ASTERICS - H2020 - 653477

Software components multi-messenger event handling

ASTERICS GA DELIVERABLE: D5.10

Document identifier:	ASTERICS-D5.10.docx
Date:	**2018-12-19**
Work Package:	**WP5**
Lead Partner:	**ASTRON**
Document Status:	**Final**
Dissemination level:	**public**
Document Link:	www.asterics2020.eu/documents/ASTERICS-D5.10.pdf

Abstract

Several instruments exist that detect cosmic events at real time, which are scientifically very interesting to follow up with observations using other instruments. Traditionally, these follow-ups need to be planned and executed manually. This planning involved manual schedule changes to the target instrument. In this document, we describe how this flow of detection, filtering, and scheduling of a follow-up observation can be fully automated, to produce an immediate response by another instrument. We provide both technical and scientific results obtained by this software.

I. COPYRIGHT NOTICE

II. DELIVERY SLIP

	Name	Partner/WP	Date
From	Jan David Mol	ASTRON	2018-10-15
Author(s)	Jan David Mol, Antonia Rowlinson	ASTRON	2018-10-15
Reviewed by	Rob van der Meer	ASTRON	2018-12-18
Approved by	AMST		2018-12-19

III. DOCUMENT LOG

Issue	Date	Comment	Author/Partner
1	2018-10-15	First Draft	Jan David Mol / ASTRON Antonia Rowlinson / ASTRON
2	2018-12-15	Final draft	Jan David Mol / ASTRON
3	2018-12-18	Final	Jan David Mol / ASTRON

IV. APPLICATON AREA

This document is a formal deliverable for the GA of the project, applicable to all members of the ASTERICS project, beneficiaries and third parties, as well as its collaborating projects.

V. TERMINOLOGY

A complete project glossary is provided at the following page:
http://www.asterics2020.eu/about/glossary/

VI. PROJECT SUMMARY

ASTERICS (Astronomy ESFRI & Research Infrastructure Cluster) aims to address the cross-cutting synergies and common challenges shared by the various Astronomy ESFRI facilities (SKA, CTA, KM3Net & E-ELT). It brings together for the first time, the astronomy, astrophysics and particle astrophysics communities, in addition to other related research infrastructures. The major objectives of ASTERICS are to support and accelerate the implementation of the ESFRI telescopes, to enhance their performance beyond the current state-of-the-art, and to see them interoperate as an integrated, multi-wavelength and multi-messenger facility. An important focal point is the management, processing and scientific exploitation of the huge datasets the ESFRI facilities will generate. ASTERICS will seek solutions to these problems outside of the traditional channels by directly engaging and collaborating with industry and specialised SMEs. The various ESFRI pathfinders and precursors will present the perfect proving ground for new methodologies and prototype systems. In addition, ASTERICS will enable astronomers from across the member states to have broad access to the reduced data products of the ESFRI telescopes via a seamless interface to the Virtual Observatory framework. This will massively increase the scientific impact of the telescopes, and greatly encourage use (and re-use) of the data in new and novel ways, typically not foreseen in the original proposals. By demonstrating cross-facility synchronicity, and by harmonising various policy aspects, ASTERICS will realise a distributed and interoperable approach that ushers in a new multi-messenger era for astronomy. Through an active dissemination programme, including direct engagement with all relevant stakeholders, and via the development of citizen scientist mass participation experiments, ASTERICS has the ambition to be a flagship for the scientific, industrial and societal impact ESFRI projects can deliver.

VII. EXECUTIVE SUMMARY

The ability to trigger instruments for follow-ups on events requires a fully automated chain from event detection to starting the follow-up observations. This need for automation creates both policy and technical challenges, in order to allow the computers to make the decisions that otherwise were made verbally through the chains of command:

- Events need to be automatically *received* from an event-detecting instrument (such as the Swift satellite or LIGO),
- Events need to be automatically *filtered* for significance,
- Events need to be automatically *translated* into specifications for the follow-up instrument.
- The follow-up observation needs to be automatically *accepted* by the follow-up instrument.
- The follow-up observation needs to be automatically *scheduled* by the follow-up instrument, possibly killing running and/or planned observations.

In this report, we describe software that demonstrates it is possible to automate this chain to connect instruments with production-level quality. We successfully allowed the LOFAR Radio Telescope to start observing in a fully automated way, after the Swift Satellite detected a possible GRB event.

Table of Contents

1. Introduction

Observing transient cosmic events with a low latency after detection, and with multiple instruments, is expected to lead to many interesting scientific discoveries. The nature of these event makes it impossible to plan observations before the event occurs, and when it does occur, typically desire data recording as soon as possible.

In this document, we report on an initial version of an example software, developed as part of ASTERICS, to fully automate the chain from receiving detected events to starting an observation in a radio telescope. Our software could be extended to listen to more events, and to request the start of observations in multiple instruments.

We present an implementation of a multi-messenger event handling system, that listens to GRB events from the Neil Gehrels Swift Observatory ("Swift")[1], and requests observations on the LOFAR Radio Telescope ("LOFAR")[2]. The telescope subsequently processes these requests against the current observational schedule, and executes the requested observation if possible.

The telescope software must be provided with enough information beforehand, in order to automate any decisions about disrupting the schedule for these ad-hoc observation requests. These mechanisms are described in this report as well.

The software we describe is fully automated and part of the current LOFAR operations.

2. Plan

We employ the following strategy.

- We recognise the parties involved in multi-messenger event handling,
- We analyse and describe their responsibilities,
- We derive interfaces between the parties involved,
- We describe and present code that provides an end-to-end implementation.

[1] See https://swift.gsfc.nasa.gov/
[2] See https://www.astron.nl/radio-observatory/radio-observatory

3. Content

Problem description

Many instruments that are capable of detecting cosmic *events* broadcast their properties, either publicly or with restricted access. These properties can include location and quality information, allowing receivers to judge whether it is scientifically very interesting to *trigger* other instrument(s), that is, point them in the direction of the event. For many use cases, fast triggering (minutes, seconds if possible) is required. A fast response demands a full automation of the control chain between event detection and the triggered instruments switching to their new target.

The problem we address in this report is what is needed to automate this process, and to show a proof of concept.

Work flow analysis

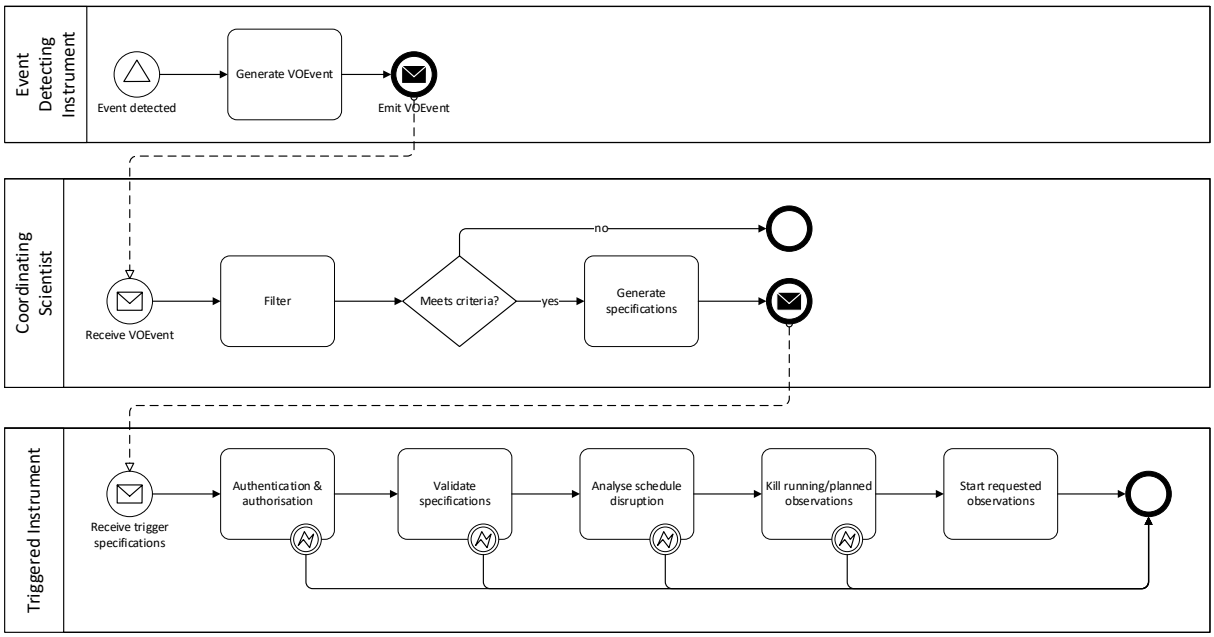
We recognise the existence of a *coordinating scientist* that is responsible for the translation of events into triggers at other instruments. This scientist is responsible for automating the receiving of the events, and filtering them on event significance and resource quotas (if any) at the instruments to trigger. Finally, it is their responsibility to translate event information into specifications suitable for each triggered instrument and scientific use case at hand.

Each instrument that is triggered will have to determine whether the *coordinating scientist* is allowed to override the running and planned schedule with their triggering request. This decision is based on several factors, all of which need to be confirmed beforehand, allowing automation at time of triggering:

- Pre-determined priority of scientist with respect to planned observations
- Resource quota given to the scientist (amount of data allowed to be recorded)
- Trigger quota given to the scientist (number of interruptions)

This document will focus on the software implementation of the event handling. The negotiation of quotas and priorities is considered out of scope.

We thus identify the following chain of actions:



Actors

We identify the following entities in our system:

- The *event-detecting instrument*, that broadcasts event messages about its discoveries,
- The *coordinating scientist* responsible for turning events into a rapid response.
- The *triggered instrument(s)*, responsible for being able to schedule low-latency follow-up observations.

Responsibilities

The *event-detecting instrument* is responsible for:

- Broadcasting event messages to the coordinating scientist.
- Annotating these events with sufficient information to determine priority and area of origin.

The *coordinating scientist* is responsible for:

- Receiving event messages from event detecting instruments.
- Filtering them on significance.

- Filtering them on applicability of the triggering instrument(s), such as its field of view.
- Translating the event to observation specifications for the triggering instrument(s).
- Broadcasting the observation specifications to the instrument(s).

The *triggering instrument* is responsible for:

- Receiving observation specifications from coordinating scientist.
- Verifying any received observation specifications against the quota of the scientist.
- Verifying the priority of the trigger against the priority of observations that will be interrupted, if any.
- Aborting running and planned observations, if needed.
- Performing the requested observation.
- Providing feedback about the processed requests.

Implementation

We provide the implementation of the event handling in two parts: software automating the workflow of the coordinating scientist, and software automating the workflow in an existing instrument to allow triggering. Both pieces of software have been developed as part of the ASTERICS project.

The following chapters will further describe these software.

4. Automating the coordinating-scientist workflow

We provide example code to automate the reception and handling of short Gamma Ray Bursts (GRBs) detected by Swift. These events are then translated and forwarded to LOFAR. An example event message as handled by this software is included in Appendix A. The message is in the common VOEvent format, and contains sufficient information to generate LOFAR specifications.

The source code is freely available at https://github.com/AntoniaR/triggerLOFAR, and works as follows. Transient events are listened for using a VOEvent Broker, which is listening to another VOEvent Broker (such as the 4PiSky Broker). When a VOEvent is received, it is first processed to check if it is one of the VOEvents users are specifically interested in. The VOEvent information is then passed to filtering code that determines if the transient event:

1. Meets the required observing criteria (e.g. duration of transient)
2. Is observable within the time requirements (e.g. above horizon and has a calibrator)

If the event passes these criteria, a trigger XML document is created using an existing template. This event is then submitted to the LOFAR systems. To do so, the submitter needs

an active LOFAR proposal, with associated username/password, and for their compute system to be whitelisted.

The above software system must be run permanently, to wait and forward events. It requires a computer system with Internet access to both the VOEvent Broker that generates the events of interest, and with Internet access to the LOFAR specification system.

Two key aspects dominate this software:

- *Correctness*: the event must provide enough information, and translation must be sufficiently good to allow the generation of a useful specification for the triggered instrument. This requires translating pointing and frequency information across domains, which could require domain-specific expertise.
- *Reliability*: not all events are worth following up, due to the event detector having a low threshold for broadcasting, and/or the limited number of opportunities for interrupting the triggered instrument. Any filtering is likely to yield both false positives and false negatives, both of which can be costly.

We therefore found that the ability to test the filtering and translation is key to writing a successful product. In our case, the coordinating scientist only gets a limited number of interrupts into the triggered instrument, which are best not wasted on the wrong events or with the wrong parameters. To facilitate this testing, LOFAR set up a test system to which scientists can send trigger requests which are validated but not executed.

As the coordinating scientist is likely to tweak their algorithms to tune both key aspects, we find it most useful to not incorporate these algorithms into the triggered instruments but explicitly require the coordinating scientist to manage these services on their own servers.

Multi-messenger triggering

Our implementation currently triggers only LOFAR, but is extendable towards triggering multiple instruments as well. To set up multi-messenger triggering, authorisation to trigger must be setup for all involved instruments, and a communication mechanism must be established for coordination.

Triggering multiple instruments can be done in two ways:

1. *Master-slave*, in which the coordinating scientist determines that instrument(s) must be triggered. Any instrument that can, does so.
2. *Coordinated*, in which the to-trigger instruments compare schedules and find the most optimal moment for a joint response.

Note that the latter approach introduced considerable complexity, and will lead to the Byzantium Generals' problem if no instrument can take authority. That is, instruments may

game the coordination to favour not interrupting their own running observations if it can get other instruments to agree.

In a more simple master-slave setup, the coordinating scientist can either directly send a trigger command to each instrument, or emit a VOEvent that (f.e.) LOFAR is being triggered, allowing independent actors at other instruments to respond in kind if possible. The complication in both methods lies mostly in divising the protocol to use in either case. Commands to trigger are highly instrument specific, and VOEvents must adhere to protocol and content rules before being broadcasted.

In this document, we chose a depth-first approach in which we explain the requirements to trigger LOFAR. Our solution can be extended towards other instruments with any of the aforementioned methods.

5. Automating the triggered-instrument workflow

We have extended the LOFAR telescope to be able to automatically receive and process observation specifications. In ASTRON, the institute operating LOFAR, we named this the *Responsive Telescope* project.

The Responsive Telescope project considered the full chain from formally requesting to be able to trigger LOFAR to delivering the status and data of such triggers to the end user. In this document, we focus on the software aspects of this process.

The LOFAR software is freely available at https://svn.astron.nl/LOFAR/trunk/. We will refer to specific parts of the source code to point at additions made for the Responsive Telescope.

In this section, we will describe a summary of the requirements gathered for the Responsive Telescope functionality in LOFAR, an analysis of LOFAR's uptime (which places hard bounds on the ability to trigger), and conclude with a description of the design of our software solution.

User Requirements

The Responsive Telescope serves scientific use cases, which give rise to several requirements as provided by our users:

1. A trigger to request for a project to:
    a. Start an observation
    b. Start an observation + processing pipeline
    c. Start an observation + processing pipeline + fast data delivery to user
2. Latency (between request arriving and action being taken): 10 s to 1 week.

3. Latency for data availability: 1 hour.
4. Ability to query the current and planned station configuration (current antenna mode, clock).
5. Ability to specify multiple SAPs in predefined patterns, depending on LBA or HBA.
6. Ability to specify constrains with respect to:
    a. Observation window (start time of window, end time of window, trigger duration)
    b. Station list (for example, use a minimum number of core stations, and/or a minimum number of remote stations)
7. Ability to track the status of the triggered observation.
8. Proposal ranking to determine if the current observation can be overrun.

The user expresses a preference for early prototyping. Phase 1 allows a latency of triggering within 30 minutes, and omits requirements 1.b and 1.c. In this document, we concern ourselves with phase 1 only.

More requirements are provided by the Radio Observatory, the department that operates the LOFAR telescope:

9. Access control to dictate when which project is allowed to trigger LOFAR.
10. Control of which observations and pipelines can be aborted or postponed in case of a fast trigger response.
11. A well-defined data delivery path to the PI must be provided.
12. Monitoring and logging of new subsystems must be implemented.
13. Monitoring and logging of events, triggers, and LOFAR's response must be implemented.
14. A response time of 5 minutes or less, using a fully automated system.
15. LTA registration of events, to allow public users to find the trigger data once it is released.
16. Software written must comply with the LOFAR standard coding practices and frameworks.

Instrument Availability

In LOFAR, the *Program Committee* decides on the priorities of the observations to be performed, and thus indirectly on the amount of time during which an observation with a specific priority can interrupt the telescope.

The LOFAR telescope has the following baseline for technical availability, that is, the amount of time it is able to run observations at all:

Reason for downtime	Planned?	Frequency	Avg duration per event	% of time
Periodic maintenance	Y	1/month	9 hours (08-17)	1.2%
LOFAR software rollouts	Y	1/6 weeks	9 hours (08-17)	0.9%
Hardware maintenance	N	2/year	24 hours	0.6%
Software maintenance	N	4/year	24 hours	1.1%
Core network unavailability	N	1/year	2 hours	0.01%
Total				3.8%

The LOFAR telescope is thus available to process triggers at >95% of the time, at the discretion of the Program Committee.

Design

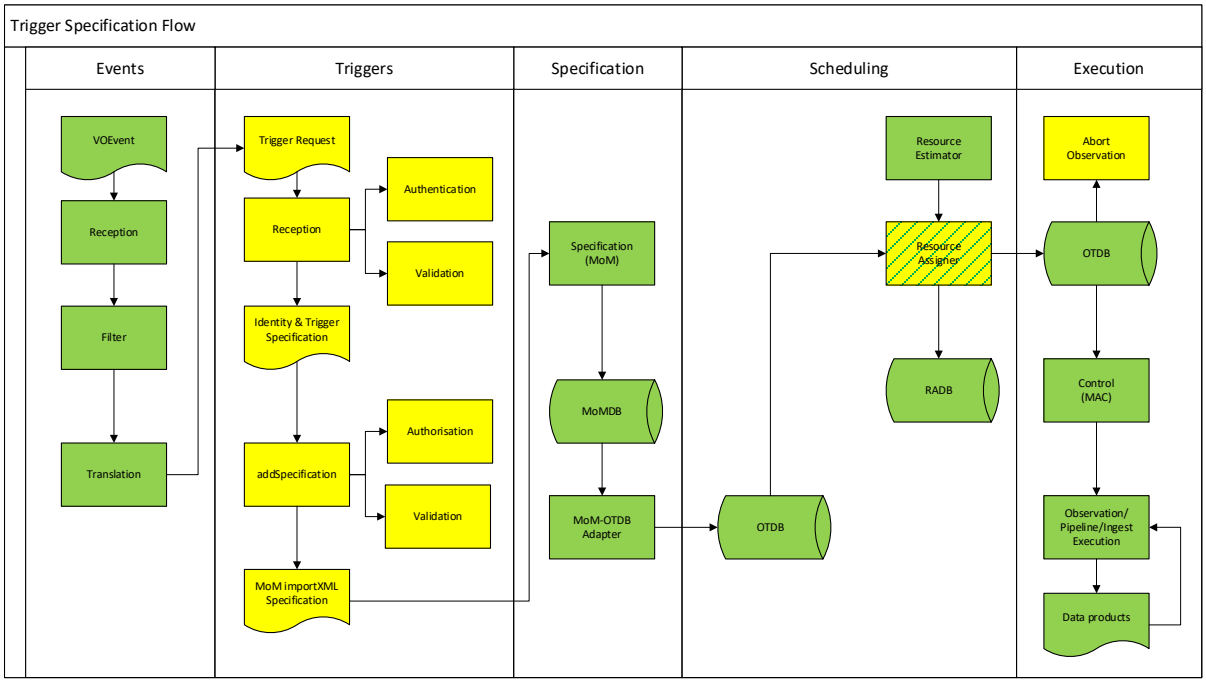
The Responsive Telescope is a feature implemented in an existing telescope, and thus relies on the existing telescope to provide most of the required functionality. We recognise the need to implement the following mechanisms:

- *Trigger handling*, that is, processing a request to interrupt the telescope with an ad-hoc observation,
- *Trigger feedback*, that is, informing the submitter of a trigger, but also the telescope operator, of the status of one or more triggers.

In the following subsections we will describe the implementation of both mechanisms.

Trigger handling

The figure below describes how new components connect to existing ones:



Existing components are shown in green, including those provided by the automation of the coordinating-scientist's workflow. New components are in yellow, and modified components are hatched. We split the workflow into 5 parts:

1. *Events*: Receiving and parsing events generated by another instrument. The responsibility of handling these is for the coordinating scientist.
2. *Triggers*: Receiving and handling of triggers. This is the bulk of implementation effort in LOFAR.
3. *Specification*: Existing systems to provide specifications for LOFAR observations.
4. *Scheduling*: Existing systems to provide scheduling for LOFAR observations. This is extended to handle ad-hoc observation requests.
5. *Execution*: Running observations. This is extended with functionality to automatically kill existing observations, if necessary to run triggers.

The triggers enter the system through a *Trigger Request*, which is received by an HTTP server serving a Django ReST interface[3]. Any request must be *authenticated* against the LOFAR user

[3] Source code: https://svn.astron.nl/LOFAR/trunk/SAS/TriggerServices/django_rest/

database, which is provided by an LDAP server. The request itself is in XML, and is *verified* by an internal LOFAR service[4] to conform to the LOFAR Trigger XSD[5].

Valid specifications are forwarded to the LOFAR specification system, through translating it to the external LOFAR Specification XSD[6], and subsequently to the internal MoM specification XSD[7]. The HTTP request returns a success (200) or failure (4xx) response based on these actions, that is, the user immediately received feedback about successful submission of his trigger into the system. After submission, the user has to rely on e-mail feedback (push), or querying the system (pull) to obtain the status of the trigger.

In most ways, the LOFAR system initially treats a trigger request as any other observation. However, at scheduling time, two or more observations might overlap in time and resources. This condition otherwise does not happen, as earlier planning stages prevent it. For triggers, the scheduling subsystem determines:

- Which observations could be killed to run the triggered observation, based on the relevant project priorities.
- Which of those it needs to kill in order to run the triggered observations, based on the conflicts in resource requirements.

If the scheduling subsystem finds a solution, it kills any observation it is required to, and in parallel, assigns resources to and requests the start of the triggered observation.

Trigger feedback

Once the trigger is released to be scheduled and executed, both the submitter and the operators will want to be informed of the status of the trigger, and to be able to query its progress. We implemented:

- Overviews of the status of triggers, provided in the same ReST interface as used for trigger submission.
- E-mailing the submitter at state changes of the trigger, such as successful submission, scheduling, and completion, or when error conditions are encountered[8].

[4] Source code: https://svn.astron.nl/LOFAR/trunk/SAS/SpecificationServices/lib/validation_service.py
[5] XSD: https://svn.astron.nl/LOFAR/trunk/SAS/XSD/SAS/LofarTrigger.xsd
[6] XSD: https://svn.astron.nl/LOFAR/trunk/SAS/XSD/SAS/LofarSpecification.xsd
[7] XSD: https://svn.astron.nl/LOFAR/trunk/SAS/XSD/MoM/LofarMoM2.xsd
[8] Source: https://svn.astron.nl/LOFAR/trunk/SAS/TriggerEmailService

6. Deviations from the plan

In the broadest sense, triggering on events in a multi-messenger fashion must involve the ability to trigger multiple instruments. We view the triggering of multiple instruments to be an extension of the triggering of a single instrument. A triggering protocol is very likely to remain highly instrument-specific to achieve the right settings of the instrument to capture the event.

The code and mechanisms we present can be extended towards triggering multiple instruments, by implementing the instrument-specific triggering commands in the code of the coordinating scientist, and by implementing instrument-specific trigger reception and handling in the triggered instrument. For both, this document describes a production-level implementation which serves as a proof of concept for other instruments.

As such, we reduced the problem at hand to the case of triggering one instrument, we believe without significant loss of generality. The provided solution can be easily extended towards optimising the triggering based on the schedule of the triggered instrument.

7. Results

We have both technical and scientific results resulting from our work. Technically, we will show the response time of the resulting solution. Scientifically, we will discuss results obtained by using the presented solution to do science.

Response Time

We measured the following delays within LOFAR for reacting on triggered observation requests. They are cumulative, and come on top of all delays between the actual event and the trigger submission:

- Triggers: 1 - 2 seconds,
- Specification & Scheduling: 11 - 40 seconds,
- Execution (Control): 60 seconds,
- Execution (Observation Execution): 7 - 31 seconds.

The LOFAR system thus expects to deliver a reaction time of a little over 2 minutes. To increase the chances of success, specifications are typically requesting to observe within 3 minutes.

The main cause of this is that the LOFAR software systems were not designed for low latencies. In later upgrades, we plan to significantly improve on the presented numbers.

Early Scientific Results

Using the software we describe in this document, Rowlinson et al.[9] triggered LOFAR to record the afterglow of GRB 180706A, an event detected by the Swift satellite. The abstract of these results are given here:

Since the first detection of gamma-ray bursts (GRBs), the community have been steadily gaining understanding of these events and their progenitor systems. Long GRBs are associated with core collapse supernovae and short GRBs occur following the merger of two neutron stars or a neutron star and a black hole. However, the nature of the central engine powering these GRBs is a subject of continuing debate with two key theories proposed: a black hole or a millisecond spin period, highly magnetised, massive neutron star (magnetar).

An observable signature of the magnetar model is a prolonged X-ray plateau phase. As accretion ends within seconds for short GRBs, the plateau phases observed are typically associated with the magnetar model. However, for long GRBs, this plateau phase has been both associated with the magnetar central engine model and with ongoing accretion onto the central engine and additional information will be required to more confidently associate these plateaus with the magnetar model.

One of the predictions of the magnetar central engine model is the presence of coherent radio emission from the newly formed magnetar, associated with the plateau phase, and this would not be present for the black hole central engine model. Thus, detection of coherent radio emission during the plateau phase of a long GRB would likely rule out a black hole central engine. There are three key coherent emission models to test:

- *persistent pulsar-like emission from the magnetar engine,*
- *fast radio bursts (FRBs) from the young, highly magnetised, neutron star,*
- *a single FRB at the end of the plateau phase if the neutron star is too massive to support itself and it collapses to form a black hole.*

However, it remains unclear if this emission is able to escape from the local and galactic environment surrounding the GRB location.

Previous studies have searched for this emission but, so far, there have been no detections. This is likely due to the small sample sizes of these studies (not all GRBs are expected to form a magnetar) and relatively insensitive searches, typically >100 Jy. One survey found a tantalising hint of two FRBs, though at very low significance, associated with the plateau phases of two long GRBs but this has yet to be confirmed. With the exception of the work by Bannister et al., which used rapid response observations by the Parkes Radio Telescope, the previous surveys have typically been either whole sky instruments (with limited sensitivity) or

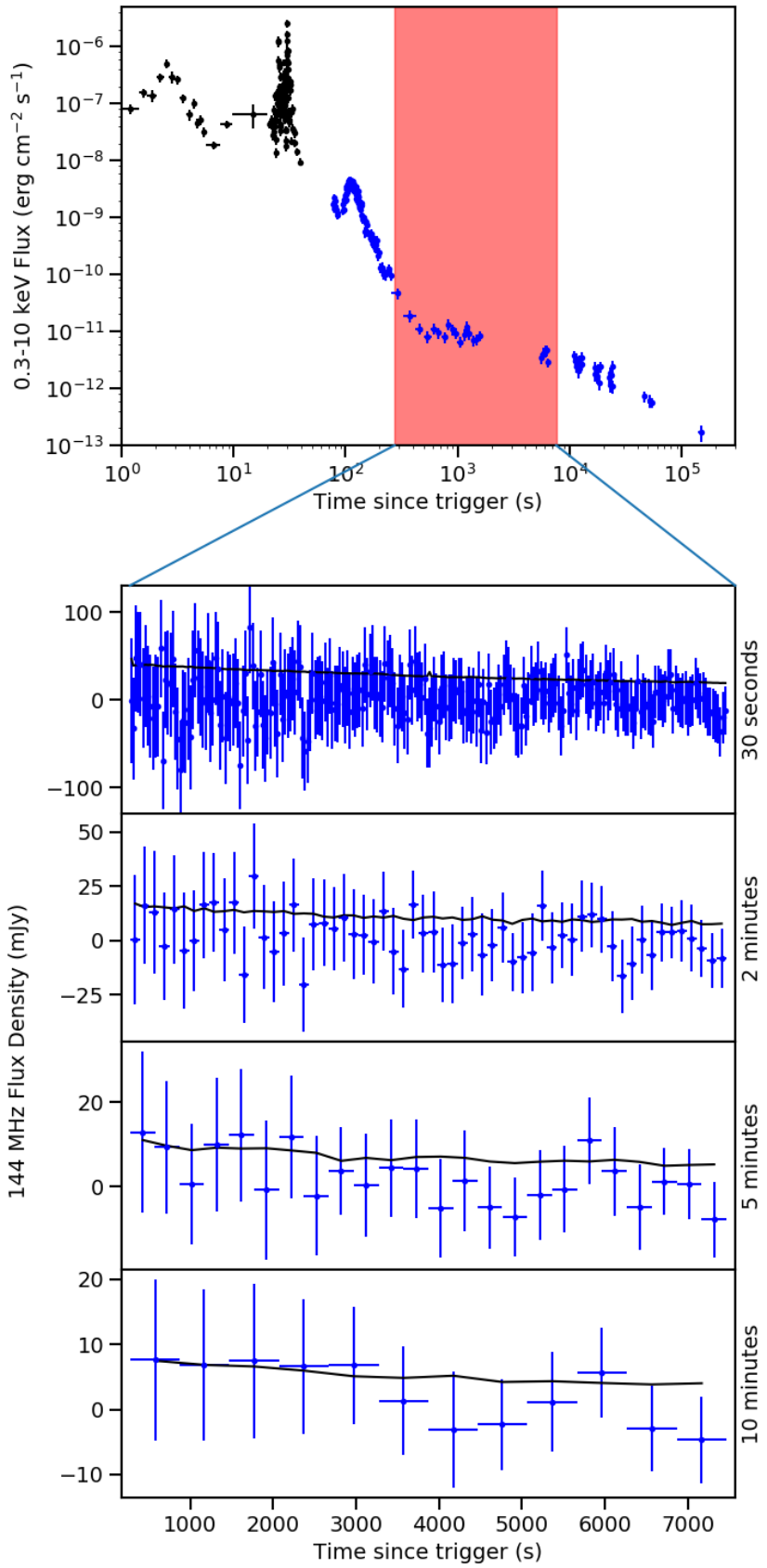
[9] *LOFAR early-time search for coherent radio emission from the central engine of GRB 180706A,* A. Rowlinson et al, Journal of Draft Publications, pp 1-9, 2018.

hampered by very slow slew times. Recently, astronomers used the Murchison Widefield Array, a low frequency radio telescope array with no moving parts, to enable a very rapid response observation of a short GRB reaching a sensitivity of ~1 Jy on 30 minute time scales. Additionally, the Long Wavelength Array, a whole sky transient survey instrument, was able to constrain prompt emission from a short GRB to a 1 σ flux density limit of 650 mJy. Although these observations are more sensitive than previous searches, they are still only beginning to constrain the emission models and deeper observations will be required.

In November 2017, LOFAR completed implementing a new rapid response mode, with observations using the full Dutch array starting within 5 minutes of receiving an alert. Although this response time is slower than that of the MWA (~30 seconds), it is sufficiently fast to study the plateau phase and there are plans to improve the response time further. By utilising the full Dutch array, a large bandwidth and a two hour observation, we can attain the sensitivity required to deeply probe for emission during the plateau phase. We successfully requested a number of rapid response triggers on GRBs detected by the Niel Gehrels Swift Observatory, and, on 6th July 2018, we successfully completed our first, fully automated, rapid response trigger on GRB 180706A.

The resulting signal is shown in the figure below. In the top panel of this figure we show the 0.3-10 keV flux light curve of GRB 180706A. The black data points were obtained by the BAT and the blue data points are from the XRT. The red shaded region illustrates the time of the LOFAR observation. In the bottom panel, we plot the 144 MHz radio flux density observations as a function of time since the GRB trigger obtained by LOFAR. We show four different snapshot time scales: 30 seconds, 2 minutes, 5 minutes and 10 minutes. The black lines in each of the LOFAR light curves are the rms noise of the images; measured from the inner 1/8th of the image.

Note that the reported 5-minute response was an early result. The minimum response time of LOFAR is currently below 3 minutes.


0.3-10 keV Flux (erg cm^{-2} s^{-1})
Time since trigger (s)
144 MHz Flux Density (mJy)
30 seconds
2 minutes
5 minutes
10 minutes
Time since trigger (s)

8. Next steps

We plan to significantly improve the response time of LOFAR during the upgrade of the relevant software subsystems. Furthermore, we plan to generalise the software for the coordinating scientist further.

9. Conclusion

The ability to trigger instruments for follow-ups on events requires a fully automated chain from event detection to starting the follow-up observations. This need for automation creates both policy and technical challenges, in order to allow the computers to make the decisions that otherwise were made verbally through the chains of command.

In this report, we describe software that demonstrates it is possible to automate this chain to connect instruments with production-level quality. We successfully allowed the LOFAR Radio Telescope to start observing in a fully automated way, after the Swift Satellite detected a possible GRB event.

Appendix A: Example Swift VOEvent

Below is an example VOEvent as generated by the Swift Satellite (anonimised):

```
<?xml version="1.0" ?>
<voe:VOEvent ivorn="ivo://nasa.gsfc.gcn/SWIFT#BAT_GRB_Pos_817345-759"
role="observation" version="2.0"
xmlns:voe="http://www.ivoa.net/xml/VOEvent/v2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ivoa.net/xml/VOEvent/v2.0
http://www.ivoa.net/xml/VOEvent/VOEvent-v2.0.xsd">
  <Who>
    <AuthorIVORN>ivo://nasa.gsfc.tan/gcn</AuthorIVORN>
    <Author>
      <shortName>VO-GCN</shortName>
      <contactName>Foo Bar</contactName>
      <contactPhone>+1-555-123-4567</contactPhone>
      <contactEmail>example@example.com</contactEmail>
    </Author>
    <Date>2018-03-24T04:37:25</Date>
    <Description>This VOEvent message was created with GCN VOE version:
1.25 07feb18</Description>
  </Who>
  <What>
    <Param name="Packet_Type" value="61"/>
    <Param name="Pkt_Ser_Num" value="1"/>
    <Param name="TrigID" ucd="meta.id" value="817345"/>
    <Param name="Segment_Num" ucd="meta.id.part" value="0"/>
    <Param name="Burst_TJD" ucd="time" unit="days" value="18201"/>
    <Param name="Burst_SOD" ucd="time" unit="sec" value="16629.63"/>
    <Param name="Burst_Inten" ucd="phot.count;em.gamma.soft" unit="cts"
value="3665"/>
    <Param name="Burst_Peak" ucd="phot.count;em.gamma.soft" unit="cts"
value="526"/>
    <Param name="Integ_Time" ucd="time.interval" unit="sec" value="0.512"/>
    <Param name="Phi" ucd="pos.az.azi" unit="deg" value="121.83"/>
    <Param name="Theta" ucd="pos.az.zd" unit="deg" value="18.04"/>
    <Param name="Trig_Index" value="124"/>
    <Param name="Soln_Status" value="0x3"/>
    <Param name="Misc_flags" value="0x0"/>
    <Param name="Cat_Num" value="0"/>
    <Param name="Rate_Signif" ucd="stat.snr" unit="sigma" value="73.98"/>
    <Param name="Image_Signif" ucd="stat.snr" unit="sigma" value="18.63"/>
    <Param name="Bkg_Inten" ucd="phot.count" unit="cts" value="17389"/>
    <Param name="Bkg_Time" ucd="time.start" value="04:36:59.32"/>
    <Param name="Bkg_Dur" ucd="time.duration" unit="sec" value="8.00"/>
    <Param name="SC_Long" ucd="pos.earth.lon" unit="deg" value="111.01"/>
    <Param name="SC_Lat" ucd="pos.earth.lat" unit="deg" value="9.01"/>
    <Group name="Merit_Values">
      <Param name="Merit_Val0" unit="dn" value="1"/>
      <Param name="Merit_Val1" unit="dn" value="0"/>
      <Param name="Merit_Val2" unit="dn" value="0"/>
      <Param name="Merit_Val3" unit="dn" value="-1"/>
      <Param name="Merit_Val4" unit="dn" value="2"/>
```

```
      <Param name="Merit_Val5" unit="dn" value="6"/>
      <Param name="Merit_Val6" unit="dn" value="0"/>
      <Param name="Merit_Val7" unit="dn" value="1"/>
      <Param name="Merit_Val8" unit="dn" value="-17"/>
      <Param name="Merit_Val9" unit="dn" value="0"/>
    </Group>
    <Group name="Solution_Status">
      <Param name="Point_Source" value="true"/>
      <Param name="GRB_Identified" value="true"/>
      <Param name="Target_of_Interest" value="false"/>
      <Param name="Target_Imag_Trig" value="false"/>
      <Param name="Target_Rate_Trig" value="true"/>
      <Param name="Def_NOT_a_GRB" value="false"/>
      <Param name="Hi_Bkg_Level" value="false"/>
      <Param name="Neg_Bkg_Slope" value="false"/>
      <Param name="Lo_Image_Signif" value="false"/>
      <Param name="VERY_Lo_Image_Signif" value="false"/>
      <Param name="Target_in_Flt_Catalog" value="false"/>
      <Param name="Target_in_Gnd_Catalog" value="false"/>
      <Param name="Target_in_Blk_Catalog" value="false"/>
      <Param name="StarTrack_Lost_Lock" value="false"/>
      <Param name="Near_Bright_Star" value="false"/>
      <Param name="Src_Removed_from_OnBoard" value="false"/>
      <Param name="Converted_SubThresh" value="false"/>
      <Param name="Spatial_Prox_Match" value="false"/>
      <Param name="Temporal_Prox_Match" value="false"/>
      <Param name="Test_Submission" value="false"/>
    </Group>
    <Group name="Misc_Flags">
      <Param name="Values_Out_of_Range" value="false"/>
      <Param name="Already_Observing_Position" value="false"/>
      <Param name="Near_Bright_Star" value="false"/>
      <Param name="Err_Circle_in_Galaxy" value="false"/>
      <Param name="Galaxy_in_Err_Circle" value="false"/>
      <Param name="ImTrig_during_ST_LoL" value="false"/>
      <Param name="TOO_Generated" value="false"/>
      <Param name="Trig_time_is_SecHdrTime" value="false"/>
      <Param name="Delayed_Transmission" value="false"/>
      <Param name="Updated_Notice" value="false"/>
      <Param name="Flt_Generated" value="true"/>
      <Param name="Gnd_Generated" value="false"/>
      <Param name="SERS_Generated" value="false"/>
      <Param name="Other_Generated" value="false"/>
      <Param name="CRC_Error" value="false"/>
    </Group>
    <Param name="Coords_Type" unit="dn" value="1"/>
    <Param name="Coords_String" value="source_object"/>
    <Group name="Obs_Support_Info">
      <Description>The Sun and Moon values are valid at the time the
VOEvent XML message was created.</Description>
      <Param name="Sun_RA" ucd="pos.eq.ra" unit="deg" value="3.21"/>
      <Param name="Sun_Dec" ucd="pos.eq.dec" unit="deg" value="1.39"/>
      <Param name="Sun_Distance" ucd="pos.angDistance" unit="deg"
value="80.00"/>
      <Param name="Sun_Hr_Angle" unit="hr" value="-4.92"/>
      <Param name="Moon_RA" ucd="pos.eq.ra" unit="deg" value="87.33"/>
```

```
      <Param name="Moon_Dec" ucd="pos.eq.dec" unit="deg" value="19.70"/>
      <Param name="MOON_Distance" ucd="pos.angDistance" unit="deg"
value="37.84"/>
      <Param name="Moon_Illum" ucd="arith.ratio" unit="%" value="44.77"/>
      <Param name="Galactic_Long" ucd="pos.galactic.lon" unit="deg"
value="152.85"/>
      <Param name="Galactic_Lat" ucd="pos.galactic.lat" unit="deg"
value="9.50"/>
      <Param name="Ecliptic_Long" ucd="pos.ecliptic.lon" unit="deg"
value="81.20"/>
      <Param name="Ecliptic_Lat" ucd="pos.ecliptic.lat" unit="deg"
value="33.69"/>
    </Group>
    <Description>Type=61: The Swift-BAT instrument position
notice.</Description>
  </What>
  <WhereWhen>
    <ObsDataLocation>
      <ObservatoryLocation id="GEOLUN"/>
      <ObservationLocation>
        <AstroCoordSystem id="UTC-FK5-GEO"/>
        <AstroCoords coord_system_id="UTC-FK5-GEO">
          <Time unit="s">
            <TimeInstant>
              <ISOTime>2018-03-24T04:37:09.63</ISOTime>
            </TimeInstant>
          </Time>
          <Position2D unit="deg">
            <Name1>RA</Name1>
            <Name2>Dec</Name2>
            <Value2>
              <C1>76.5845</C1>
              <C2>56.7246</C2>
            </Value2>
            <Error2Radius>0.0500</Error2Radius>
          </Position2D>
        </AstroCoords>
      </ObservationLocation>
    </ObsDataLocation>
    <Description>The RA,Dec coordinates are of the type:
source_object.</Description>
  </WhereWhen>
  <How>
    <Description>Swift Satellite, BAT Instrument</Description>
    <Reference type="url" uri="http://gcn.gsfc.nasa.gov/swift.html"/>
  </How>
  <Why importance="0.90">
    <Inference probability="0.90">
      <Name>GRB 180324</Name>
      <Concept>process.variation.burst;em.gamma</Concept>
    </Inference>
  </Why>
  <Description></Description>
</voe:VOEvent>
```