Improving IVS Communication through a VLBI Communications Center

M. Bérubé[1], J. Gipson[1], J. Lovell[1], D. Lakins[2]

Abstract Since 2000, the International VLBI community has greatly invested in developing new hardware and software to improve data precision, processing, and latency. There is also a significant effort in automating observations and processes. Unfortunately, one element that has not been improved is how the different VLBI components, Operation Centers, stations, correlators, and analysts are communicating. The current VLBI communication system was developed 25 years ago and relies mainly on emails, archiving system, and the dedicated people monitoring the information relevant to them. This is not a suitable communication system for operational VLBI.
To improve the actual system, the GSFC group at NASA has developed a VLBI Communications Center (VCC) and tools for near real-time, machine-to-machine, two-way communication between IVS components. The VCC is a web service supported by a database and a message broker using formatted information designed for access by computers. The database keeps up-to-date data on schedules, catalogs and all relevant information on various IVS components (e.g., station's availability, latest SEFDs). The message broker is used to inform any IVS components that some data/information at the VCC are relevant for them. The VCC knows who acknowledges the message and uploaded the schedule, allowing full traceability of data/information exchanges. This paper will show the limitations of the current communication system, describe the various components of the VCC, and how it could improve the communication and data exchange between IVS components. The tests done with the VCC hosted by NASA will also be presented.

Keywords IVS, VLBI, VCC

1. NVI, Inc./NASA Goddard Space Flight Center, 7257 Hanover Parkway, Suite D, Greenbelt, MD 20770, USA
2. NASA Goddard Space Flight Center, Greenbelt, MD 20771, USA

1 Introduction

Many IVS components have automated their processes but still need human interactions when communicating with other IVS groups. IVS components are exchanging information through email exploders and Data Centers. Many delays are built into processes to account for uncertainties in data synchronization and communication exchanges.

Because of IT security issues, it is not possible for IVS components to open their IT infrastructure to other groups to facilitate data exchange. To bypass this issue, a central dedicated server, the VLBI Communications Center (VCC), was developed to provide near real-time, machine-to-machine, two-way communication system between all IVS components.

2 Current Method of Communication

The current communication method was developed 25 years ago around IVS mail exploders, archiving systems, and web pages.

The IVS mail exploders are sending hundreds of emails every week to registered members on different issues. People need to spend time reading these emails to evaluate their relevance and extract valuable infor-

mation. Unless people reply to these emails, there is no easy way to know if messages have reached the target audience. Some applications were developed to extract valuable information regarding master schedules and station operational data with limited success. These emails are written for people and are not following a standard format that could be easily read by software.

The IVS Data Centers are mainly archiving systems for safekeeping of the IVS data files. Because of their operational and synchronization rules, it may take minutes to hours before a submitted file is available at all Data Centers. This synchronization issue may create situations where different versions of a file reside on the Data Centers at the same time. The way files are retrieved from Data Centers, it is not possible to know who has downloaded specific files and when.

This is not an operational system for rapid exchange of information and traceability. Some delays are required to ensure the specific IVS components have received the information and retrieve the needed data files. For example, schedules must be submitted seven days in advance to avoid problems inherent in this communication method. Schedules are rarely modified if stations cannot observe as expected to avoid the use of different schedules.

3 Proposed VLBI Communications Center

The VCC was designed for removing uncertaintains in IVS data exchange and communications and supporting automation. The main characteristics of the system are:

- Fully automated;
- Machine-to-machine data exchange;
- Near real-time, two-way communication; and
- Traceability of data/information exchange.

The VCC has three essential components needed to support all its functionalities:

1. Database 
2. Web Service (API) 
3. Message Broker 

3.1 Database

The main purpose of the database is keeping the latest information on sessions, catalogs, station performances (SEFDs), and availability (maintenance) for more efficient scheduling. All transactions, messages, and events at the VCC are kept in the database for traceability. For example, we may need to know who has downloaded or uploaded files and when to ensure that everyone has the same version of a schedule.

All VCC users must be registered and have access to specific functionalities. The database keeps credentials on each user along with its roles and responsibilities. For example, only the Operation Center identified in the master schedule could upload the schedule for a session.

The database is populated and queried using the Web Service.

3.2 Web Service

The VCC Web Service is an application programming interface (API) that conforms to the constraints of REST architectural style. The users provide or retrieve data through http requests (get, post, put, or delete) using specific JavaScript Object Notation (JSON) data exchange format. Users provide credentials using Jason Web Token (JWT) included in the header of the http requests. The API is validating all users and ensure the request is valid for this user. The API provide credentials to access the message broker. After answering a request, the API analyzes who should know about this and informs specific users through the message broker.

3.3 Message Broker

The VCC is a near real-time, two-way communication system but cannot connect to other servers to deliver messages. The VCC uses a message broker dispatching structured messages using specific rules. Messages are routed to one or more queues that store information until a user acknowledges having received them. Appropriate routing rules are dynamically created so the users are receiving only messages relevant to them.

Each IVS component has its specific queue or inbox. Users need to connect to the message broker to retrieve messages in their inbox.

4 VCC Data Flow

To show how the VCC could deliver information in near real-time, Figures 1 to 5 illustrate the process of uploading a schedule and making sure every station has it. This approach could greatly reduce the submission time for schedules.

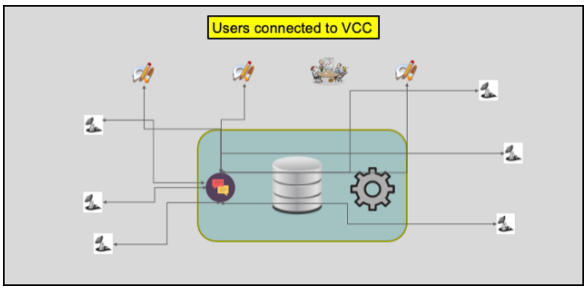

Fig. 1 Users connected to VCC. The VCC is not pushing any messages to external servers, so users must be connected to their private inbox to receive messages.

The Operation Center (OC) needs to generate a new schedule at pre-defined time prior to the start of a session. The OC could query the VCC to have the latest information on station performance and availability. After generating the new schedule, the OC uploads it to the VCC API (Figure 2).

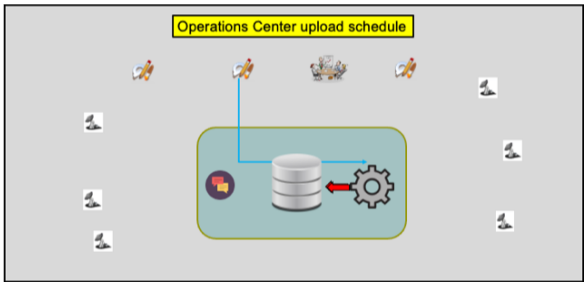

Fig. 2 Operation Center submits new schedule. First, the API validates that the OC is resposible for this session. After uploading the file, the VCC logs the process in the Database and informs the OC of success.

At any time, the API could be used to retrieve the status of a session. The returned information includes the time the last schedule was uploaded and when the stations downloaded it.

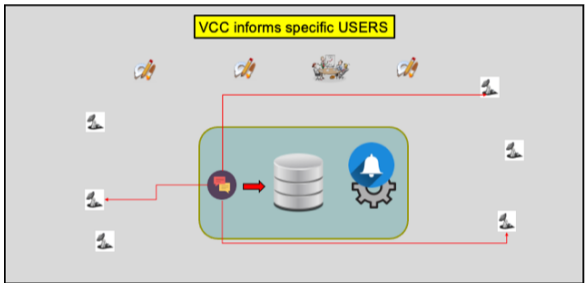

Fig. 3 VCC informs specific users. The API generates a message and dispatching rules to inform specific users through the message broker. If registered, Correlators and Analysis Centers would receive the message.

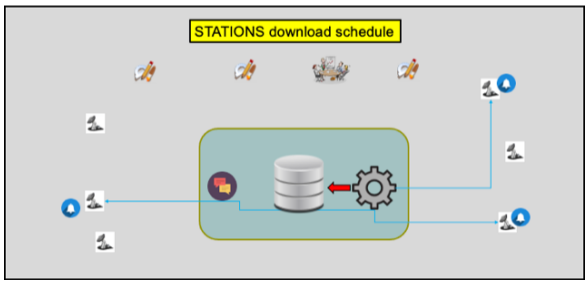

Fig. 4 Stations download schedule. On receiving the message, stations download the schedule file from the VCC and could drugd it. The VCC is recording who downloaded the file and when.

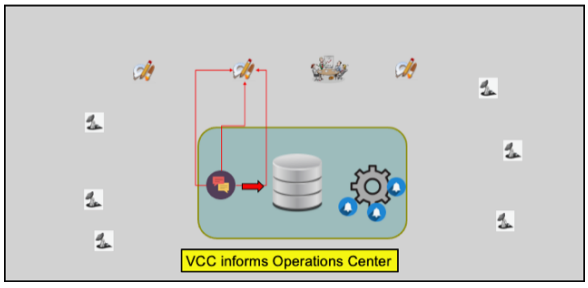

Fig. 5 VCC informs Operation Center.

This process took few seconds with the maximum delay coming from stations downloading the schedule. If something is wrong, a new schedule could be safely uploaded ensuring that all stations are using the same version.

5 Proof of Concept

To demonstrate how IVS components could communicate and exchange information through a VLBI Communications Center, we developed a first version of the VCC server and client applications for Coordinating Center, Operation Centers, and Network Stations.

A limited number of messages were created for the first demo.

A Dashboard application (Figure 6) was developed to show the exchange of information between the different IVS components. The Dashboard uses the API to get the latest status of a session and stations. It also connects to the message broker to receive any message related to the session. This tool could be used by any IVS component to check on session status.

5.1 VLBI Communications Center

- Linux server (Ubuntu)
- MariaDB
- Web Service
  - Python using FastAPI
  - Data exchange using JSON
  - Trigger most events for informing IVS component
- Message broker
  - RabbitMQ
  - Message structure has been defined
  - Dispatching rules have been defined

5.2 VCC Client Applications

- Coordinating Center
  - Create/update sessions
  - Maintain catalogs (database)
- Operation Center
  - Retrieve data (network availability, performence, catalogs) for generating schedule
  - Submit schedule (skd, vex, txt) files
- Network Station
  - Message listener
    - sessions have changed (get session information from API)
    - new schedule (download schedule, drudg)
  - Message publisher to inform IVS of station status (e.g., scan, source, problems)
  - Data uploader for SEFDs or logs
- Dashboard
  - Session viewer that could be used by any IVS group. See Figure 6.

The first demo was simulating a four-station schedule lasting ten minutes with a problem at one of the stations. The station application was not running at existing stations but on different servers than the VCC. Using the VCC message broker, one of the stations informed IVS that it could not participate due to power outage at the site. The Operation Center uploaded a new schedule that was downloaded by all participating stations in few seconds. The Dashboard showed when stations downloaded the new schedule and started observing. It also displayed the antennas downtime and recording status (e.g., scan, source) during simulated observation. During the demo, two Dashboards were running at the same time at different locations showing that VCC configuration is automatically adapting the rules for dispatching messages to all interested users.

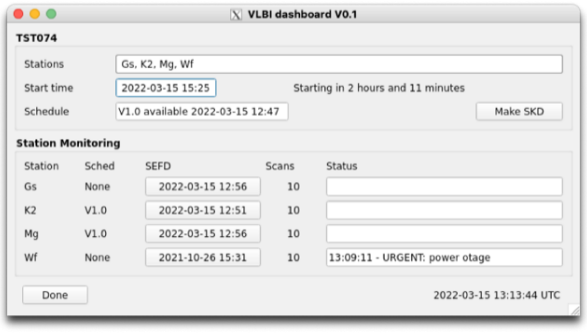

Fig. 6 VLBI Dashboard.

6 Conclusions

The actual IVS communication method needs to be improved to remove uncertainties, reduce delays in information exchange, and facilitate automation. The VCC has shown that it is possible to have a near real-time, machine-to-machine, two-way communication system between IVS components without compromising IT security.

The actual VCC is near operational mode, but we still need to define/add more messages and functionalities to cover most of the IVS activities. The VCC has a very flexible design and can easily accept more tasks as needs increase.

A Python package and applications were developed to help the different IVS components to access the VCC to submit or retrieve information.

The VCC is intended for IVS users with specific roles and restrictions. To limit access to IVS users only, the latest VCC is accessed through SSH tunnel and users need to submit a public SSH key. For extra security, some information between users and VCC API are encrypted using the SSH keys and Jason Web Token (JWT).

Documentation for accessing VCC API and Message Broker will be developed for Python and non-Python users.