Web-based MUSER Remote Monitoring Data Visualization Method Postprint

Authors: Zhou Xinlei, Wang Feng, Shoulin Wei, Mei Ying, Liu Cuiyin

Date: 2017-09-26T00:00:00+00:00

Abstract

Astronomical remote monitoring constitutes a crucial component of astronomical research. To promptly detect anomalous data arising from weather conditions, electromagnetic interference, antenna malfunctions, and other factors, and to enhance the reliability of data processing, the Mingantu Spectral Radioheliograph in China urgently requires the implementation of real-time remote monitoring of observation status. This paper proposes a visualization method for remote monitoring data based on WebSocket technology, validated through real-time display of power diagrams and spectrograms during observations. Experimental results demonstrate that the method satisfies the requirements for real-time monitoring data visualization, transforms the traditional on-site monitoring paradigm, eliminates geographical limitations, and offers numerous advantages including high reliability, excellent real-time performance, and broad scalability, thereby improving work efficiency and equipment utilization, and providing valuable reference for other remote monitoring data visualization systems.

Full Text

A Web-Based Visualization Method for MUSER Remote Monitoring Data

Zhou Xinlei[1], Wang Feng[1], Wei Shoulin[2], Mei Ying[1], Liu Cuiyin[1]

[1]Key Laboratory of Computer Technology Applications of Yunnan Province, Kunming University of Science and Technology, Kunming, Yunnan 650500, China
[2]Yunnan Observatories, Chinese Academy of Sciences, Kunming, Yunnan 650011, China
Email: wf@cnlab.net

Abstract

Astronomical remote monitoring constitutes an important component of astronomical research. To promptly detect anomalous data caused by weather conditions, electromagnetic interference, and antenna malfunctions, thereby improving data processing reliability, China' s Mingantu Ultrawide Spectral Radio Heliograph (MUSER, originally named Chinese Spectral Radio Heliograph–CSRH) urgently requires implementation of remote monitoring for real-time observation status. This paper proposes a visualization method for remote monitoring data based on WebSocket technology, validated through real-time display of power and spectrum diagrams during observations. Experimental results demonstrate that the proposed method satisfies the requirements for real-time monitoring data visualization, transforms traditional on-site monitoring modes, eliminates geographical constraints, and offers numerous advantages including high reliability, excellent real-time performance, and broad scalability. The method enhances work efficiency and equipment utilization, providing valuable reference for other remote monitoring and data visualization systems.

Keywords: Remote real-time monitoring; WebSocket; Visualization

1. Introduction

To advance solar radio observation technology, Chinese astronomers proposed the Mingantu Radio Spectral Heliograph (originally named Chinese Spectral Radio Heliograph, CSRH), which enables simultaneous high spatial, temporal, and frequency resolution observations of the Sun in the centimeter band to better study solar dynamic properties [1]. The Mingantu Radio Spectral Heliograph represents a crucial component of the National Astronomical Observatories' Mingantu Station. MUSER consists of two synthetic aperture arrays: MUSER-I comprises 40 4.5-meter antennas imaging at 40 frequencies (0.4–2 GHz), while MUSER-II consists of 60 antennas imaging at 60 frequencies (2–15 GHz). With hardware construction largely completed and the instrument entering operational observation phase, astronomers currently control observation equipment on-site to acquire, process, and analyze observation data. Each observation session requires physical presence at the facility, which proves inconvenient for astronomical observations. The ability to conduct observations remotely without on-site presence has become a current priority, as it enhances research efficiency while reducing observation costs. Remote automated observation benefits the development of astronomical observation technology, and the prerequisite for achieving this capability lies in obtaining and visualizing remote real-time observation data. Research on implementing remote real-time monitoring data visualization holds practical significance for acquiring and visualizing remote observation data and carries major importance for realizing automated observations.

2. Current Status of Astronomical Remote Monitoring Visualization

Traditional network application information exchange follows the HyperText Transfer Protocol (HTTP) pattern, where clients send requests through browsers, servers process these requests after reception and verification, return result information to client browsers, and browsers subsequently render the results. This method suffers from low real-time performance and can no longer satisfy the demands of high-rate astronomical observation in the data-intensive era. Maintaining real-time, rapid information synchronization between client and server represents a critical challenge for current astronomical observations.

In the astronomical domain, polling technology has been fundamentally adopted to achieve remote real-time communication. Polling involves browsers sending requests to servers at specific time intervals, with servers returning the latest data to client browsers [2]. This traditional request pattern exhibits obvious drawbacks: browsers must continuously send requests to servers, request headers are lengthy, consuming considerable bandwidth. Comet represents a newer polling-simulation technology that achieves full-duplex communication to some extent but remains inefficient, still requiring requests that waste substantial time and network resources while demanding robust server support, thereby burdening servers and severely impacting real-time network communication.

Faced with this situation, WebSocket—a new protocol defined in the HTML5 specification—makes remote real-time visualization of astronomical telescope monitoring data possible [3]. It implements full-duplex communication between browsers and servers. When the WebSocket protocol establishes full-duplex communication, browsers and servers require only a single handshake action. WebSocket establishes a fast channel between browser and server, enabling direct bidirectional data transmission. This simple yet efficient communication mechanism breaks traditional HTTP request-based communication patterns, allowing servers to actively push data to browsers and maintain information synchronization between server and browser. The communication headers in WebSocket' s connection establishment process consume minimal resources compared to traditional HTTP requests, conserving bandwidth. WebSocket' s full-duplex communication mechanism and efficient communication performance provide robust technical support for remote real-time visualization of astronomical telescope monitoring data [4].

3. Implementation of Remote Monitoring Data Visualization

3.1 Remote Monitoring Data Visualization Requirements The overall framework for MUSER remote monitoring is illustrated in [Figure 1: see original paper]. Observation data received by digital receivers undergoes correlation preprocessing, with one portion sent to high-performance distributed clusters for later imaging and solar activity analysis, while another portion serves for real-time monitoring of raw observation system data. Observation data is typically affected by multiple factors such as electromagnetic interference, antenna track-

ing accuracy failures, and system gain anomalies from data reception equipment, all of which cause observation data abnormalities. The data monitoring system serves as an important basis for marking anomalous data during data processing and constitutes a crucial guarantee for reliability in later data analysis.

[Figure 1: see original paper] Mingantu Radio Spectral Heliograph Remote Monitoring Overall Framework Diagram

According to observation requirements, observation data from both low and high frequency arrays must be visualized in real-time at certain intervals (e.g., 1-minute intervals for the low-frequency array). Previous work has achieved on-site visualization of observation data, with a multi-screen display prototype system for on-site observation data visualization already operational [5]. In on-site data visualization, data is transmitted from data reception servers to data monitoring servers via the network, then displayed on screens after visualization processing. However, on-site data visualization display does not facilitate remote visualization monitoring of observation data. To effectively enable remote real-time monitoring of observation data, a network-based remote data visualization system must be designed [6]. Data processed periodically by the data processing system is sent via network to a network visualization server, enabling clients to perform remote visualization of observation data through web browsers [7].

Currently, mainstream browsers support WebSocket protocol for full-duplex communication. Therefore, only the network server side requires WebSocket protocol support. Visualization network servers process data through visualization processing and send it to browsers for visualization display via the WebSocket protocol.

3.2 Visualization Network Server Design and Implementation

3.2.1 WebSocket Network Framework Selection Among numerous current network frameworks, CherryPy—a Python-based, object-oriented framework that maps URLs to Python callable objects—was selected as the visualization server-side framework technology. CherryPy is simple and easy to use. The ws4py library implements WebSocket protocol support for CherryPy, providing a high-level, encapsulated, and simple interface. User applications need only inherit the WebSocket class and override methods according to application requirements to implement WebSocket protocol support for network applications.

3.2.2 Network Server Implementation The server class relationships are shown in [Figure 2: see original paper]. The ws4py WebSocketHandle class provides an interface implementing the handshake protocol. The WebSocketHandle class offers two virtual functions, send() and received_message(self, message), enabling simple implementation of network application business logic. In concrete implementation, the WebSocketHandle class is inherited to implement

business logic, with the received_message() method handling each request' s business logic before forwarding the message.

[Figure 2: see original paper] WebSocket Server Class Diagram

The ws4py WebSocketPlugin class inherits and implements the CherryPy SimplePlugin abstract interface, which defines the plugin registration interface for the CherryPy framework. The Engine class inherits and implements the WebSocketPlugin API. The entire framework' s runtime management is built upon a single bus instance. To cooperate with the bus, the WebSocketPlugin defines WebSocket protocol support interfaces. The bus API function declarations are as follows:

```
cherrypy.engine.publish(channel, *args)
cherrypy.engine.subscribe(channel, callable)
cherrypy.engine.unsubscribe(channel, callable)
```

To establish a WebSocketHandle instance as a plugin, the plugin instance defines the msgbroadcast() method for handling each request' s business logic. After processing each request' s business logic, this method forwards the requested message. The msgbroadcast() method achieves broadcasting functionality by registering with the CherryPy Engine.

3.2.3 Remote Visualization Process for Monitoring Data The process for implementing remote monitoring of observation data through the remote visualization framework in the Mingantu Radio Spectral Heliograph is illustrated in [Figure 3: see original paper]. The browser sends a request to the network server. After the server and browser establish a WebSocket handshake connection, the network server obtains observation data from the data acquisition interface for visualization processing. The network server then sends the processed visualization data to the browser for display via the WebSocket protocol.

[Figure 3: see original paper] MUSER Visualization Process Diagram

3.3 Visualization of Monitoring Data To achieve remote real-time visualization of antenna auto-correlation and cross-correlation data, this paper designed and implemented a remote visualization prototype system for auto-correlation and cross-correlation diagrams, displaying these data in real-time through remote networks. To enable multi-dimensional analysis of antenna data, an auto-correlation data spectrum graphical display method was designed and implemented. By monitoring auto-correlation spectrum data of visibility data, the system displays auto-correlation data in real-time through network browsers.

The auto-correlation spectrum structure design is shown in [Figure 4: see original paper]. Based on the designed structure and related calculation methods, the QtGui library' s QImage painting class object QPixmap draws the auto-correlation spectrum as shown in [Figure 5: see original paper]. In the auto-

correlation spectrum diagram, the X-axis represents data frame numbers, the Y-axis represents power values, and each curve represents one antenna. The power values of 16 channels for each antenna are displayed, with each channel' s power value represented by one color. The frequency values of each antenna channel can intuitively determine antenna status information: normal antennas exhibit uniform frequency amplitude changes, while abnormal antennas show large frequency amplitude variations or zero amplitude values.

[Figure 4: see original paper] Auto-Correlation Spectrum Structure Diagram
[Figure 5: see original paper] Auto-Correlation Frequency Spectrum Diagram

3.4 Network Visualization Display of Monitoring Data By entering the URL address in the client browser to connect to the visualization network server, the network server processes source data sent by the data acquisition system in real-time and performs visualization. The system displays auto-correlation and cross-correlation data, meeting MUSER' s requirements for real-time visualization of monitoring data. Cross-correlation data network visualization display is shown in [Figure 6: see original paper], auto-correlation data network visualization display in [Figure 7: see original paper], and auto-correlation spectrum data network visualization display in [Figure 8: see original paper].

[Figure 6: see original paper] WEB Visualization of Cross-Correlation Data
[Figure 7: see original paper] WEB Visualization of Auto-Correlation Data
[Figure 8: see original paper] WEB Visualization of Auto-Correlation Frequency Spectrum Data

4. Performance and Results

Utilizing real-time technology, remote visualization display of observation data can visualize one frame of visibility data in approximately 1.5 seconds, generating 16 cross-correlation diagrams and 40 auto-correlation spectrum diagrams. Each image generation requires about 0.0375 seconds, far less than MUSER' s data processing system interval of one frame per second, satisfying remote real-time visualization requirements for the data monitoring system. Under LAN conditions, generating one frame of data (16 cross-correlation diagrams and 40 auto-correlation spectrum images) and displaying them in web browsers via WebSocket protocol from network servers requires only 2.8 seconds, while under WAN conditions requires approximately 4.3 seconds, meeting remote visualization real-time requirements for monitoring data. MUSER observation data can be displayed in real-time through web browsers on desktop and laptop computers, facilitating observation personnel to analyze and judge data anytime and anywhere. This capability clearly benefits remote real-time monitoring of observation data.

5. Conclusion

This paper describes in detail a method for remote monitoring and visualization of observation data obtained by the Mingantu Radio Spectral Heliograph using web technology. WebSocket technology enables real-time, rapid rendering of visibility data correlation diagrams and auto-correlation spectrum diagrams. The web-based remote monitoring data visualization display demonstrates high efficiency: generating 16 cross-correlation diagrams and 40 auto-correlation spectrum images for network page display requires only 0.5 seconds, meeting remote monitoring system real-time requirements. The system prototype has been deployed and operational at the Mingantu Station, achieving excellent application results. The web-based development process is relatively simple, with completed code being cross-platform and producing consistent display effects, breaking geographical constraints on real-time monitoring data processing and analysis. This facilitates timely detection of anomalous observation states and observation target activities, providing reference for telescope real-time monitoring and large antenna array status monitoring, and maximizing scientific output.

References

[1] Chen Zhijun et al. Progress on Chinese solar radioheliograph in cm-dm wavebands. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 91–98.

[2] Yan Yihua, Zhang Jian, Chen Zhijun, et al. WebSocket application solution. Computer Knowledge and Technology, 3826–3828.

[3] Weng Zhaosong, Yi Renwei, Yao Hanbing. WebSocket based real-time Web application solution. Computer Knowledge and Technology, 3826–3828.

[4] Wei Shoulin, Cao Zihuang, Wang Feng, et al. A study of web control of an RTS2 system based on the WebSocket. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 404–409.

[5] Wessels A, Purvis M, et al. Remote data visualization through websockets. Jackson J. New Generations 2011 Eighth International Conference. 2011: 1050–1051.

[6] Zhou Xinlei, Wang Wei, Wang Feng, et al. Design and implementation of a multi-monitor display system based on QT for NAOC MUSER observations. Astronomical Research & Technology, 503–509.

[7] Han Weijie, Zhang Wen, Li Xiaomei. Design and implementation of Web-based visualization system. Computer Engineering, 218–220.

[8] Zhang Wen, Li Xiaomei. Research and implementation of Web-based visualization. Computer Engineering & Science, 25–27.

Note: The original manuscript contained several formatting artifacts and fragmented text segments that have been consolidated into coherent academic prose while preserving all technical content, figure references, and bibliographic information.

Note: Figure translations are in progress. See original paper for figures.