Conception of a computer monitoring and function-specific remote controlling application framework

József Bór[1]* and Csongor Szabó[1]

[1]Institute of Earth Physics and Space Science (ELKH EPSS), Sopron, Hungary

Abstract

An idea is described for the implementation of a framework by which the operation of a computer can be monitored and controlled remotely. Main advantages of the proposed solution are that (1) it can be implemented using open source code and applications, (2) remote monitoring and control can be made using a general web browser, (3) the operation does not need large bandwidth internet connection, (4) it gives system administrators complete control over the production, format, and appearance of the content that serves monitoring purposes, (5) remotely controllable operations can be fully customized, and (6) the proposed remote controlling solution does not make the supervised computer more vulnerable for hacker attacks. The proposed application framework can be an optimal solution for monitoring and remote controlling computers that manage data acquisition from different measurements or run programs which require occasional user intervention.

Keywords: remote control, computer management, operation monitoring, content sharing, automatization.

Introduction

The possibility of monitoring and controlling a computer remotely has many practical advantages, e.g., checking proper operation of the system and programs, supervising the activity of users, providing help, sharing information, or simply working without the need of being physically near the hardware. Users usually want as complete access as possible to the functionalities and settings over which they possess rights in a specific system. For system administrators, this means complete control over the computer while for normal users, this means the same regarding the range of services available within their accounts.

*Corresponding author: József Bór (bor.jozsef@epss.hu)

The most straightforward concept of remote monitoring/accessing a computer is direct access to the system over the internet. In this model, a service (that provides access) is running on the computer and clients connect directly to the system through this service. Several applications and communication protocols have been developed in this sense to allow remote access to a computer (Comparison of remote desktop software, 2021). Advanced applications can be set to allow partial access to the system to limit the risk of malicious usage. This means either allowing solely view-only access or providing possibilities to use/monitor only specified applications. This is called application sharing (Taylor, 2020).

Applying this solution has several pitfalls. Opening direct access to a computer always raises the risk of becoming the victim of hacker attacks. The service itself must be configured carefully. Service programs use optimized data transfer methods and often proprietary communication protocols so that the user does not know how these programs work and has limited control on what information is transmitted. Setting up direct access may require additional configuration if the computer is located behind a firewall, e.g., port forwarding (How to Port Forward, 2021). This is not always easy (or can be even impossible) to realize when the user has no rights to configure the firewall (e.g., in a company or university). Note that the issue regarding firewalls can be solved by tunneling the transferred information through an independent web service (Tunneling, 2021). The maintenance of such servers, however, is costly so that such solutions are usually provided as a paid service. This solution may not be suitable for users having trust concerns because the operation depends on a third party server and all transmitted information is channeled through such a server. Several service applications have their corresponding specific client software the installation of which may also cause inconveniences under different operating systems (Comparison of remote desktop software, 2021).

Another approach concerns mostly content-sharing needs, therefore it can be considered as a monitoring solution. This model, too, requires a service installed on the system. In this case, the task of the service is to upload or stream the content to be shared to a server and users/viewers can reach the information by connecting to the server rather than to the content-providing (CP) computer. This solution is safe because it does not rely on incoming requests to the CP system. Firewall issues are rare because the upload is initiated from inside by the information-providing system. Nevertheless, difficulties in setting up such solutions may come up because installing and configuring a server correctly as well as creating the corresponding user interface requires specific knowledge. Well established content-streaming applications and corresponding web-based services are available for live-sharing popular content such as audio and video (List of streaming media systems, 2021). These solutions are optimized to these content (media) types and may generate considerable data traffic. They often require wide band internet connection. The nature of this approach does not allow direct control of the CP system.

Here we conceptualize the framework which allows either monitoring and/or controlling a computer remotely, is relatively easy to set up, eliminates firewall issues, and is not content-specific. The motivation of considering the development of such a solution is triggered by the need for being able to monitor the operation of different measurements running in the Széchenyi István Geophysical Observatory (IAGA code: NCK) (Bór et al., 2020) and, occasionally, changing some settings as required. Another specific field of possible application is enabling co-workers and citizen scientists to remotely supervise and manage a computer-controlled optical observation system built for observing upper atmospheric electro-optical phenomena. The monitoring part consists of obtaining and organizing text and images of different formats from different sources/programs in the CP system regarding the various measurements. On the other hand, controlling of the recording systems may consists of starting and stopping different applications, changing settings and making various complex actions (minimizing/maximizing windows, mouse clicking buttons, capturing the screen, etc.). This should be achieved without allowing a user to have complete control over the full functionality of an application or the system. One or few applications that handle all these tasks could be coded and users could be allowed to fully control only these applications using application sharing solutions. We propose a more universal and transparent solution to this problem so that the content to be remotely seen and the actions that can be initiated remotely can be completely and easily customized.

Fundamentals of the proposed framework

In the proposed framework for remote monitoring and management of computers, the content-providing (CP) system is separated from the content-sharing (CS) system. The CP system is a computer, possibly behind a firewall, while the CS system is supposed to be a public web server. Remote users connect to the web server with a general web browser and see the uploaded actual information or may interact with the web server if the content is interactive. The information on the web server is kept up to date by the CP system, from where the latest information is uploaded to the server regularly. In the implementation of controlling functionality, the CP system also queries the web server for user requests.

Content management on the supervised computer

One key part of the proposed solution is a content manager (CM) routine that is scheduled to run periodically on the CP system. The routine completes a few main tasks. The order of completion of the tasks can be arbitrary, nevertheless, keeping the following order is recommended. One main task can be querying the web server for requests from the remote user. If there are requests, the routine starts the execution of the corresponding actions. The routine also starts the execution of code and commands which provide output to be shown for the remote user (monitoring functionality). The routine then refreshes the content on the CS system.

The CM routine can be launched regularly using the task scheduler service that is usually available on the operating system, or can be embedded in a program loop within the CM application. The first solution is preferred if the goal is only monitoring and the information of interest is refreshed rarely. Using the system's task scheduler imposes too much load on the system if the content needs to be refreshed frequently. If the usage is of such, setting up the CM routine in a program loop is preferred. It is also easier to start and stop a single application than to configure system-level settings or services. Note that a single application, too, can be configured to run as a service so that continuous operation of the application can be guaranteed if needed.

In a practical implementation, the CM routine runs the CP code and applications by cycling over batch files in a single directory. The order of the names of the batch files can determine the order of execution. This way, the actions can be overviewed easily, the order of execution can be changed simply by renaming the batch files appropriately, and tasks can be created or removed simply by adding or deleting batch files, respectively. This also has the advantage that the CM framework does not need to be edited directly.

Care must be taken not to start programs that run for a long time or can freeze and block the running of the CM routine. It is suggested that long running or unstable programs be started as a separate process in the background. It can be a good practice to insert a code in the beginning of the corresponding batch file that checks if the previously started program is still running. If the process is still running, it can be either killed before it is run again or starting it can be skipped either completely or at least for a given time-out interval.

It makes it easier for the CM routine to manage the content produced by the applied code and programs if the output of those applications is directed to a single directory. It is suggested that the content is to be produced in ‘web-ready’ format so that the CS system on the web server does not need to process it further and so the remote user can view the information as it gets uploaded. Another advantage of this implementation is that the content can be checked also locally. Note that local installation of a web server may be necessary for a local user to be able to verify the functionality of the content if it has interactive parts. Including active code in the web content is highly suggested so that the information in the remote user’s browser can be automatically refreshed (ajax methods) (Zhaoyun et al., 2010; Faang, 2021) without the need for manual re-loading the page. Note that browsers or corporate firewalls may store previously downloaded web content in a cache (Caching Behavior of Web Browsers, 2014). This may cause that, for example, an image does not change in the remote user’s browser although it has already been refreshed on the web server. If the name of the updated image remains the same, the browser may not recognize the change and keeps reloading the latest image from the cache instead of the new file on the server. This behavior can be avoided by passing a changing parameter (e.g. current time) along with the file within the HTML code (Lutkevich, 2020) so that the browser recognizes the change and always downloads the new file from the web server.

Especially if the monitored content is composed of several parts, changes quickly, and the refresh rate is high, producing the monitored content comes with many disk I/O operations. Initiating many disk I/O requests can not only cause slow program operation, but it also speeds up wear out of the hard disk (Wu et al., 2018). In case of such requirements, and if the size of the content as well as the available computer memory (RAM) makes it feasible, it is suggested to use a virtual RAM disk (RAM Disk, 2021) for collecting the newest version of the monitored content.

On the other hand, the internet traffic corresponding to uploading the files to the CS web server can be lowered by keeping track of the files in the monitored content and by uploading only those that are new or have changed. This can be implemented by creating verification codes of the uploaded files on the web server using a hashing algorithm (CRC, MD5, etc.) (Basatwar, 2021). By comparing those codes to the corresponding codes of the actual files on the CP system, synchronization of the content on the web server can be limited to the changed information: uploading new files, deleting those that are not present anymore, and replacing those that have changed.

Handling remote requests on the supervised computer

If remote control is needed, it is a task of the CM routine to query the CS system on the web server for new requests from the remote user and to manage the completion of those requests. This functionality can be part of the duty cycle for the monitoring part but it can also be implemented independently. The advantage of the latter is that querying the web server is not restricted to the refreshing interval set for the monitoring CM routine. Instead of that, the 'long polling' technique (Faang, 2021) can be used which maintains a communication channel between the managed computer and web server. The same technique can be applied regarding the communication between the remote user and the web server. This results in quick refreshing of the information in the remote user's browser whenever the content on the web server is changed. Regarding the controlling functionality, requests from the remote user can be instantly received and processed as well as simple responses can be sent back to the remote user via the CS system if this technique is applied.

To keep track of the status of the remote requests, it is suggested that they are to be downloaded to a specific folder on the controlled system (e.g., a folder named 'requests_received') with an attached identifier (e.g., timestamp) so that the time order of their execution can be unambiguously determined. The requests themselves are supposed to be text messages consisting of a command word and optional arguments. For each requestable valid command, a batch file can be created (supposedly with an identical name) so that the CM routine can easily parse the requests and will handle only those that have a corresponding executable batch file. The CM routine should cycle over the downloaded requests. If a request is valid, the CM code runs the corresponding batch file with the requested arguments and copies the request entry to another folder named, e.g., 'requests_processed' along with its direct return value (text). Note that the concerns about long-running and hanging processes should be considered when the request-processing batch files are coded. Also note that it is completely valid for request-processing programs to also create output in the folder where the content produced by the system-monitoring scripts is collected. That way, the corresponding results can appear in the monitoring section of the application, too.

Finally, the CM routine should upload the content of the folder 'requests_processed' to the web server so that the remote user can follow the course of actions. If the upload completes successfully, the requests can be archived for bookkeeping purposes in another folder named, e.g., 'requests_archived'. That way, the received requests, their order of execution and their direct simple return messages can be reviewed simply and easily any time locally on the supervised computer.

Creating and registering new commands the execution of which can be requested can be implemented in a fairly user-friendly way. The CM routine can be coded to check the list of request-processing batch files regularly. This way, the validity of the received requests can be checked against the actually available list of functionalities. This strategy ensures that only valid requests are accepted from the remote user in a way that service files on the CS system need not to be modified at all.

The suggested solution ensures that remote operation is fairly safe as only those commands can be run remotely for which the implementing batch file exists on the controlled system.

It may be worth considering adding a security feature to the code on the web server to ensure that routines there accept content only from the CP system. A simple solution for that is to pass a security code invisibly as part of all POST messages (Williams, 2021a) to the CS system on the web server. The validity of that code can be checked and any communication will be started only if the code is okay. Communication can be further secured by using secure HTTP protocol (HTTPS) (Friyanto, 2019; Williams, 2021b). When HTTPS is used, the information is transferred in an encrypted form so that it cannot be reconstructed even if it is captured.

Functionality of the code on the content-sharing system

The code on the CS system (web server) has three main roles. It receives the content to be shown from the CP system, it manages the communication with the remote user (i.e, it receives controlling requests), and it passes the received requests to the CP system upon being queried. If the content of the monitoring functionality is produced in a web-ready format, the code on the CS system does not need to process it further. The only task related to those files is the creation of their verification code via the selected hashing algorithm.

There are several details which are worth considering in the implementation of these functionalities. For instance, it can be advantageous from the point of view of compatibility in the communication if the code on both the CP and the CS systems is implemented using the same software framework (e.g., Python, PHP, Java Script, etc.). Regarding the handling of the requests of the remote user, it is suggested to set up a folder structure similar to that on the CP system. Remote requests are suggested to be saved in a folder named, e.g., 'requests_received', together with a timestamp. After these requests are transferred to the CP system, they can be moved to another folder named, e.g., 'requests_sent' so that, depending on the implementation of the user interface, it is possible for the remote user to follow their status. Once the requests are processed on the CP system, prompt results (ideally bundled with the original requests) are suggested to be uploaded into another folder named, e.g., 'request_processed' and then the corresponding entries in the 'requests_sent' folder can be removed. The user interface should be able to show to the remote user both the actual status of the requests and the prompt simple messages they returned.

If the monitoring/controlling functionality is to be used by a single user, simple password protection of the access to the shared content and to the controlling interface may provide a sufficient level of security. Opening the usage of especially the controlling interface for multiple users requires additional considerations. To avoid confusion, it is suggested to allow only for one user at a time to control the CP system. The related functionality must be managed by the code on the CS system. A conception of a possible solution for handling multi-user access is given below.

Each user is supposed to have a unique username and a corresponding password. Once one of the users allowed to control the CP system signs in, its IP address would be registered. For that user, a controlling session starts so that the login interface is not further provided for other users who visit the login page from IP addresses different from that of the signed-in user. The time point of gaining control or the time point of the latest request sent (whichever is later) is recorded for the active user. If that user is inactive for a specified amount of time, the IP lock gets released so that other allowed users may log in and take the control over. It is advised that there be a supervisor who can always sign in and can take the control over. In case of multi-user implementations, it is suggested for bookkeeping purposes that the username be recorded together with each request sent.

Conception of a flexible content presenting framework

The proposed framework for a monitoring and remote controlling solution allows the administrator of the supervised computer to set up both the shared content itself and the appearance of the content on the web server completely freely. Nevertheless, designing and implementing a user-friendly web page which can handle automatic refreshing of the displayed content, not to mention the user interface for controlling requires specific knowledge. We propose a basic framework for the displayed content itself which can be easily used by an average computer user to set up any basic set of status-monitoring information.

The main idea behind the conception is that each bit of information to be shared has its own simple HTML code snippet which determines the appearance of the corresponding content (text, images, etc.). One or more of such code snippets together with its corresponding files (e.g., image files), if needed, can be generated by each batch file (or the process started by it) run by the CM routine on the CP system. The final web page containing all information is supposed to be compiled by a master code which incorporates all actually existing code snippets. The order of appearance of the content in the web page can be the alphabetical order of the filenames containing the HTML code snippets. This way, the user can determine the appearance of each part of shared information freely and independently from other content entries.

Summary

We described the conception of a general framework for remote monitoring and controlling the operation of a computer. The proposed solution gives the administrator of the supervised computer full control over both the content to be presented for monitoring purposes and its appearance. The framework is based on the idea that the remote user is connected to a web server where the presented content for monitoring and, optionally, the user interface for remote control is available (Fig. 1). The content on the web server is updated regularly by a content-manager routine running on the supervised computer. It causes no problem if the CP system is working behind firewalls since all communication between the web server and the supervised computer are initiated from the supervised computer and can be implemented using common HTTP or HTTPS protocols.

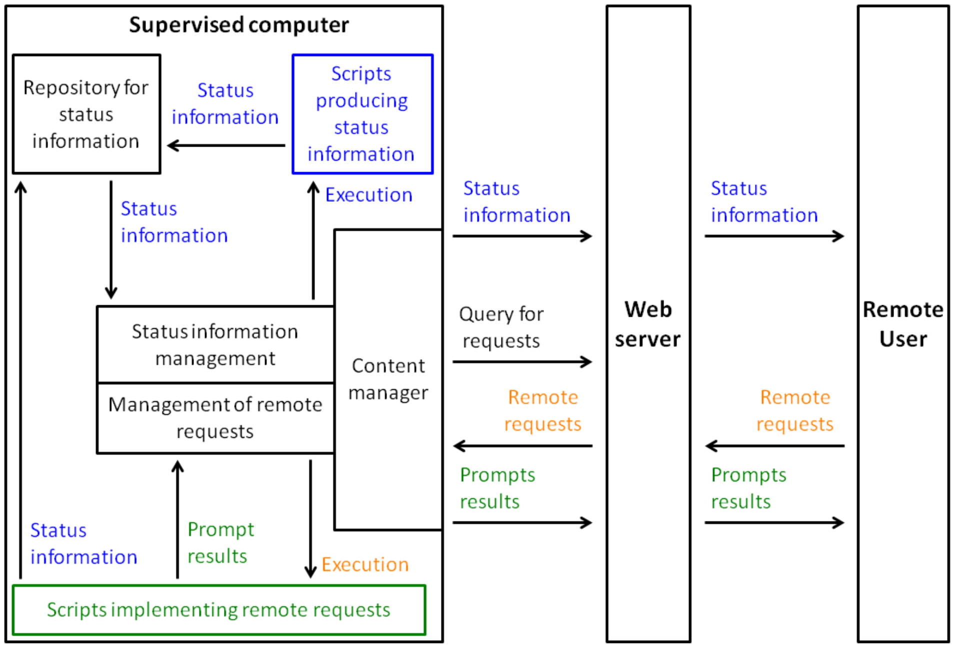

Fig. 1: Flow chart of the proposed monitoring and controlling application framework.

The framework may include the functionality of making specific operations on the supervised computer remotely controllable without opening any general purpose access channel to the supervised system. The proposed solution makes it possible that the set of remotely controllable options, functions, and services in the supervised computer can be fully and easily customized, reviewed, and managed by the administrator of the system. The framework itself, therefore, can be considered very safe from the point of view of hacker attacks from the internet. Note that it is the responsibility of the administrator of the supervised system to allow only safe operations to be remotely requestable.

Finally, a general framework was suggested for easy management of various types and appearances of contents to be shown to the remote user via the web server. The main idea behind that framework is that files containing HTML code snippets are produced by each monitoring content-provider code or program and a master routine compiles these code snippets into a complete web page that appears in the browser of the remote user. Note that no parts of the proposed application framework require seamless internet connection. Should any communication attempt fail, it will be re-tried in the next refresh cycle.

Beside describing the general ideas, several aspects of the possible implementation are discussed including security-related questions and options for extending the framework to be used by more remote users. Realization of the proposed conceptions may uncover additional problems that need to be solved for the system to work optimally. Nevertheless, we are confident that a communication system that works basically according to the given guidelines performs well for the intended purposes.

Acknowledgements

This work was supported by the National Research, Development and Innovation Office, Hungary (NKFIH; grant No. K115836).

References

Basatwar, G. (2021). Hashing Algorithms – An In-Depth Guide To Understanding Hash Functions. Retrieved November 30, 2021, from https://www.appsealing.com/hashing-algorithms/

Bór, J., Sátori, G., Barta, V., Szabóné-André, K., Szendrői, J., Wesztergom, V., Bozóki, T., Buzás, A., & Koronczay, D. (2020). Measurements of atmospheric electricity in the Széchenyi István Geophysical Observatory, Hungary. *History of Geo- and Space Sciences, 11*(1), 53-70, https://doi.org/10.5194/hgss-11-53-2020

Caching Behavior of Web Browsers (2014). Retrieved November 30, 2021, from https://www.f5.com/services/resources/white-papers/caching-behavior-of-web-browsers

Comparison of remote desktop software (2021). Retrieved November 30, 2021, from https://en.wikipedia.org/wiki/Comparison_of_remote_desktop_software

Faang, C. (2021). Ajax polling vs long-polling vs WebSockets vs server-sent events. Retrieved November 30, 2021, from https://medium.com/geekculture/ajax-polling-vs-long-polling-vs-websockets-vs-server-sent-events-e0d65033c9ba

Friyanto, A. (2019). Hyper Text Transfer Protocol for Securing Packet Inspection in Intrusion Prevention System Device. *IOP Conference Series: Materials Science and Engineering, 662*, 022021, https://doi.org/10.1088/1757-899X/662/2/022021

How to Port Forward – General Guide to Multiple Router Brands (2021). Retrieved November 30, 2021, from https://www.noip.com/support/knowledgebase/general-port-forwarding-guide/

List of streaming media systems (2021). Retrieved November 30, 2021, from https://www.wikiwand.com/en/List_of_streaming_media_systems

Lutkevich, B. (2020). HTML (Hypertext Markup Language). Retrieved November 30, 2021, from https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language

RAM Disk (2021). Retrieved November 30, 2021, from https://www.techopedia.com/definition/2801/ram-disk

Taylor, N. (2020). The Collaboration Series: Chapter 4 – What Is Application Sharing? Retrieved November 30, 2021, from https://www.gamma.co.uk/resources/blog/what-is-application-sharing/

Tunneling (2021). Retrieved November 30, 2021, from https://www.speedcheck.org/wiki/tunneling/

Williams, L. (2021a). GET vs POST: Key Difference between HTTP Methods. Retrieved November 30, 2021, from https://www.guru99.com/difference-get-post-http.html

Williams, L. (2021b). HTTP vs HTTPS: What is Difference Between HTTP and HTTPS? Full Form. Retrieved November 30, 2021, from https://www.guru99.com/difference-http-vs-https.html

Wu, S., Yi, Y., Xiao, J., Jin, H., & Ye, M. (2018). A Large-Scale Study of I/O Workload's Impact on Disk Failure. *IEEE Access, 6*, pp. 47385-47396, https://doi.org/10.1109/ACCESS.2018.2866522

Zhaoyun, S., Xiaobo, Z., & Li, Z. (2010). The Web asynchronous communication mechanism research based on Ajax. *2nd International Conference on Education Technology and Computer*, pp. V3-370-V3-372, https://doi.org/10.1109/ICETC.2010.5529524