August 2022

Network Status Management and Networkless Logging

Theodore Obbard

David Phillips

Bhawana Sharma

Network Status Management and Networkless Logging

ABSTRACT

This disclosure describes techniques of networkless logging of client application events when the client device does not have an internet connection. Networkless logging as described herein is request-agnostic, stores requests locally (e.g., on device memory or disk), and, if a network connection is presently unavailable, sends out the requests at a later time when the network connection becomes available. This disclosure also describes a network status manager that can accurately and reliably obtain the state of the network (connected or disconnected) by sending fast, lightweight requests to a remote endpoint that is geographically close to the user.

KEYWORDS

- Networkless logging
- Network availability
- Queries per second
- Polling
- Navigator API
- Content delivery network (CDN)
- Network status manage

BACKGROUND

Some video hosting and sharing websites enable users to download videos to their client devices and watch them later to enable users to watch videos without an active internet connection. However, without the client device having an internet connection when the user is watching a video, requests and logs sent from the client device to the server are lost.

A related problem is the accurate, reliable, and fast detection by the client device of the presence or absence of a stable network connection. Although there are tools such as the JavaScript navigator API that report network connection status, such tools typically perform only a superficial check for network connectivity, e.g., by reporting the presence or absence of a WiFi connection. No automatic pinging to a server to verify end-to-end network connectivity is performed, likely due to a perceived communications burden on the client device.

DESCRIPTION

This disclosure describes techniques of networkless logging of a client application without the client device having an active internet connection. Operative on a client device, networkless logging as described herein is request-agnostic, stores requests locally (e.g., on device memory or disk), and, if a network connection is unavailable, sends out the requests at a later time when the network becomes available. The entire request is stored post-packaging, such that a wide variety of logging requests from virtually any client application can be served, and logs can be sent to virtually any endpoint.

This disclosure also describes a network status manager that can accurately and reliably obtain the state of the network (connected or disconnected) by sending fast, lightweight requests to an endpoint that is geographically close to the user.

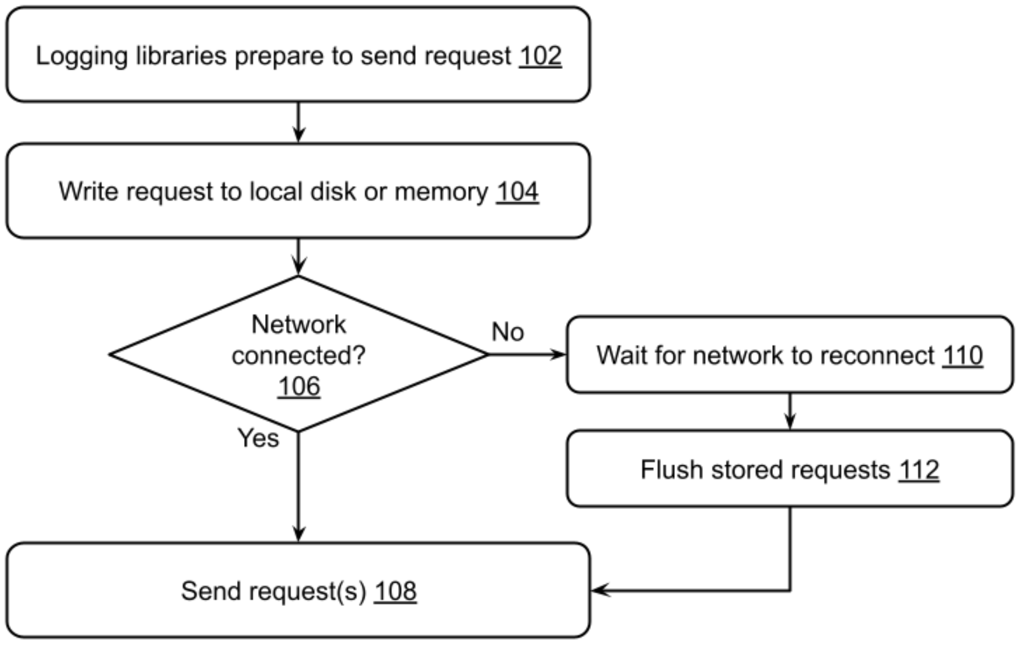
Logging libraries prepare to send request 102
Write request to local disk or memory 104
Network connected? 106
No
Yes
Wait for network to reconnect 110
Flush stored requests 112
Send request(s) 108

Fig. 1: Networkless logging

Fig. 1 illustrates networkless logging. Logging libraries running on a user device prepare to send one or more requests (102). The request is written to local disk or memory (104). A check is made regarding whether a network connection is active (106). If there is an active network connection, the one or more requests are sent (108) to the remote server, e.g., that provides videos. If no active network connection is detected, a wait state is entered (110). Once a network connection becomes available, the stored requests are flushed (112) - sent over the network to the server.

Networkless logging, as described herein, is made request-agnostic by ensuring that it has as few dependencies as possible. Any application that has a need for networkless logging, e.g., video sharing app, mapping/navigation app, conferencing app, etc. can utilize it. On-device storage can take the form of JavaScript Object Notation (JSON) objects, local storage provided by JavaScript, database, disk storage libraries, etc. Stored data is structured by user, such that user data is compartmentalized.

Each request stored in on-device storage is stored with certain keys, as follows:

- The URL and options object: These keys comprise the actual request and are used to send the request once a network connection becomes available.
- The timestamp, which enables the sending of requests in order, e.g., most recent first (LIFO) or any other order.
- A status field of either "NEW" or "QUEUED" The status field prevents multiple tabs from trying to send the same request, as follows:
    - When a request is first added to the storage, it is marked as "NEW." Once it gets to the sending procedure, it is marked as "QUEUED," preventing another tab

from pulling it out of storage. An atomic read-and-write is used when marking stored requests as QUEUED.

- A request that has been successfully sent is removed from on-device storage.

The aforementioned keys on each request reduce the juggling of states on the client application and ensure that the appropriate operation runs on the appropriate set of requests upon the network becoming available.

As indicated in Fig. 1, for networkless logging to operate, the status of the network (connected or not) is to be determined. Network connectedness can be determined by a network status manager, which can be a library that utilizes a variety of mechanisms to maintain the most accurate and current state of the network connection. While using conventional sources of network connectedness status such as the JavaScript navigator API, the network status manager validates network changes by sending a fast, lightweight request to a remote endpoint that is geographically close to the client device. The network status manager can also use polling to check if the network has been regained by trying to send a validation request at suitable intervals, e.g., every thirty seconds. If the network is unavailable, these checks (which are in the form of pings or queries) are free of cost, since they incur zero queries per second (QPS) at the server and occupy zero network bandwidth, since queries do not reach the server. On the other hand, when the network becomes available, the polling reports a connected network the first time it reaches a server, such that the QPS burden on the server remains low. To reduce network traffic, once online, the network status manager can retest for network connectedness when specifically requested by a client application.

To further reduce network traffic, the network status manager pings an endpoint geographically close to the user, not necessarily the server of the video sharing provider or other content delivery server that the client application normally interacts with. The endpoint accessed by the network status manager can be at the edge of a cloud provider, such that no deep-cloud traffic is generated by a large volume of devices sending frequent pings. The endpoint can advantageously have no other function other than to respond to pings. To spread the load of responding to pings, there can be multiple ping-responding servers that are geographically distributed, each responding to pings generated by devices geographically close to them. In contrast to the network status manager, the networkless logging application connects to the remote video sharing or other server within a content delivery network.

The network status manager provides application programming interfaces (APIs) which can be called by an application to check if an active network connection is available; to indicate that a network connection is (or isn't) available; etc. Informing the network status manager that the network connection is (or isn’t) available is a form of user override. The network status manager can advantageously be set up as a singleton class, such that it is accessible to multiple applications that run on a client device and returns the same, consistent answer to all applications.

The described networkless logging techniques, implemented with specific user permission, enable users to view videos while offline, improve reliability (where timeliness is not a factor) and quality of viewership metrics, and are applicable for any request that is not user facing. The described techniques ensure that the logs of a user that watches a video on their phone in a train that goes through a tunnel, thus causing a temporary loss of network connection, are not lost. Although described in the context of video sharing, networkless logging is applicable for any client application that connects to a remote server and for which timeliness (of requests and logs) is not a factor. The network status manager can be used by any client-side code or application, e.g., a navigation/mapping app, an audio/video conferencing app, content viewing and/or content sharing app, etc. that seeks to know the status of the network connection.

CONCLUSION

This disclosure describes techniques of networkless logging of client application events when the client device does not have an internet connection. Networkless logging as described herein is request-agnostic, stores requests locally (e.g., on device memory or disk), and, if a network connection is presently unavailable, sends out the requests at a later time when the network connection becomes available. This disclosure also describes a network status manager that can accurately and reliably obtain the state of the network (connected or disconnected) by sending fast, lightweight requests to a remote endpoint that is geographically close to the user.

REFERENCES

[1] Chang, Ching-Jye, and Lorin Evan Ullman. "Method and system for network management with adaptive monitoring and discovery of computer systems based on user login." U.S. Patent Application Number 09/738,337 filed Dec 15, 2000.

[2] Steven M. Cohn, "Detect network availability," https://www.codeproject.com/Articles/64975/Detect-Internet-Network-Availability accessed July 29, 2022.

[3] John Au-Yeung, "Checking network status with the network information API," https://levelup.gitconnected.com/checking-network-status-with-the-network-information-api-2021d9dc6a7e accessed July 29, 2022.