March 2020

Client-device Telemetry Using System Privileged Application

Anonymous

Client-device Telemetry Using System Privileged Application

ABSTRACT

This disclosure relates to a telemetry application that reliably logs events on a client device with accurate timestamps, enables an enterprise that controls the client device to turn data collection on/off on a per device basis, and runs as a system-level process of the device operating system. The use of a single, system-level telemetry application allows early load of the application on the client device which saves time when loading individual applications. It also provides a single source of updates and changes. The telemetry application provides a single pipeline for telemetry data allowing gathering of statistics and source data for events.

KEYWORDS

- Telemetry
- Virtual reality (VR) headset
- Augmented reality (AR) device
- Event logging
- Timestamp correction

BACKGROUND

Client devices such as virtual reality headsets, smartphones, game consoles, etc. are commonly used for purposes such as playing games, participating in virtual environments, etc. One important aspect of use of such devices is the logging of various events that take place on the client device, known as telemetry. It is important that events be recorded reliably, with accurate timestamps and in the correct sequence with minimal overhead.

DESCRIPTION

This disclosure relates to a telemetry application that reliably logs events on a client device with accurate timestamps, enables an enterprise that controls the client device to turn data collection on/off on a per device basis, and runs as a system-level process of the device operating system. The use of a single, system-level telemetry application allows early load of the application on the client device which saves time when loading individual applications. It also provides a single source of updates and changes. The telemetry application provides a single pipeline for telemetry data allowing gathering of statistics and source data for events.

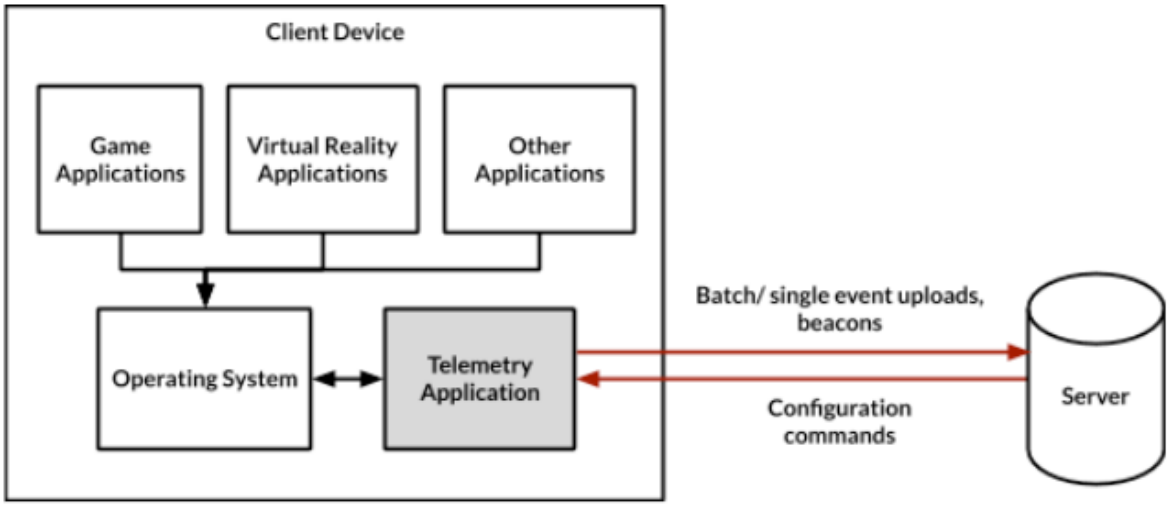

Fig. 1: Telemetry application

Fig. 1 illustrates an example client device with a telemetry application. The client device runs various applications such as game applications, virtual reality (VR) applications, and other applications on an operating system. Per techniques described herein, a telemetry application is provided on the client device that enables reliable data collection from the client device. The collected data is transferred over a network to a server via batch uploads and/or uploads of single

events. The server can send configuration commands to the client device, e.g., to enable/disable logging of various types of events.

The telemetry application runs as an application with system level privileges which allows access to system libraries and files, network, thermal information, memory statistics, audio interfaces, battery, etc. The telemetry application can be accessed as a service on the OS and runs in a persistent manner, e.g., is not killed when the client device starts to run out of memory, is re-launched automatically on crash, and is launched at boot time.

The telemetry application standardizes the payload that is delivered to the server, e.g., by using a common header, which helps with data analysis and validation and enables matching of different events within a session from the same client device. The telemetry application can also ensure that personally-identifiable information (PII) is removed from the data from the client device, per user settings. The telemetry application can categorize events and control each category with specific settings, e.g., related to how these are sent to the server. For example, categories can include device health, application performance, user-driven events, high priority events, etc. The telemetry application stores events in memory and uploads events to the server, immediately or in batch mode, based on the category of the event. The telemetry application provides a single, central telemetry pipeline.

A particular problem with event logging is when the client device is used with factory set time, e.g., without connecting the client device to a network such as WiFi that enables updating the device time to the current time using network time protocol (NTP). Events that are logged in such devices have incorrect timestamps based on the local time of the device and thus, such data is unreliable for querying. Further, if the timestamp is not updated prior to upload, the server

may drop such data if it is deemed too old. In some cases, all such data may be assigned the same timestamp (e.g., time of the batch upload) which causes the sequence of events to become unavailable.

The telemetry application as described herein addresses these issues by automatically correcting the timestamp. The real-time clock (RTC) time of the device is a monotonically increasing time value for the lifetime of the device, and is unaffected by device reboot or factory reset operations. Events that occur prior to the time being set are stored in a buffer. The telemetry application determines whether the time has been set for the device, e.g., via NTP. IF the time is set, events that were previously recorded with RTC time and stored in the buffer are retrieved and their timestamps are corrected based on the current RTC time and current device time. This ensures that such events are not dropped and are available for querying.

The telemetry application also provides a single point to log events from different applications. By having a single pipeline, an enterprise can control logging from the telemetry application, e.g., to select specific types of data for upload, to turn specific types of logging on or off, add standardized data to events, etc. This helps prevent loss of data and upload of bad data.

The telemetry application also enables detection of data loss that can be caused by network connectivity issues, crashes, or server-side errors. Data loss can be detected at batch-level (e.g., one or more events from a batch upload) and at event level (a single event that was sent by the client device, but not received at the server). This can be accomplished by periodically transmitting beacons that include metrics about events that were sent in the prior period.

CONCLUSION

This disclosure relates to a telemetry application that reliably logs events on a client device with accurate timestamps, enables an enterprise that controls the client device to turn data collection on/off on a per device basis, and runs as a system-level process of the device operating system. The use of a single, system-level telemetry application allows early load of the application on the client device which saves time when loading individual applications. It also provides a single source of updates and changes. The telemetry application provides a single pipeline for telemetry data allowing gathering of statistics and source data for events.