MULTI-LAYER DATA-PLANE TELEMETRY COLLECTION DESIGN FOR SYTEMS WITH MULTIPLE OBSERVERS

Supriya Kannery (Dell Technologies), Kalyan Gunda (Dell Technologies)

ABSTRACT

Existing telemetry-related technical discussions focus on how to efficiently collect data from multiple systems for a single observer. In this paper, the focus is on a generic telemetry design for a single system observed by multiple distinct observers. For example, in a server, a management application collects capacity usage-related data while the vendor's telemetry application collects hardware-related parameters, license-related data, etc. Multiple observers can have both common and unique data set used for observability. As the frequency of collection is varied across observers, telemetry collection design needs to have approaches to a) avoid “observer effect”, b) use hybrid data collection methods like streaming and pull. We present and discuss the pros and cons of different architectures for the use case of multiple observers for telemetry collection. Our proposed design uses REST API -based telemetry architecture with multi-layer data planes.

KEYWORDS

Telemetry, Data-plane, Observers, Multi-layer, Collection

INTRODUCTION

Multiple observers collecting data from a single source is a prevalent scenario. Examples from different areas like Aviation, Healthcare, and Servers are explained in this paper. System performance and behavior get impacted as multiple observers collect data and transfer it to observers. This is known as the observer effect. In this paper, a generic design is proposed that can address the observer effect for systems that need multiple observers.

The main content of this paper has 3 sections. Section 1 brings up real-life scenarios where multiple observers collect data from distinct sources in a single entity and shows how prevalent this scenario is. Section 2 focuses on the challenges faced when multiple observers collect required distinct and varied data from a single server. Section 3 proposes a generic design to address major challenges in a server and explains its pros and cons. Section 4 compares proposed architecture with existing architectures.

1. MULTIPLE OBSERVERS AND DATA COLLECTORS

1.1 HEALTHCARE

Different IoT devices collect data and share it with respective device vendors. Fitness bands collect steps taken and share that with a mobile application or respective observer. Simultaneously, implanted devices for tracking heartbeats or blood pressure collect related data and sharing with the respective observer. In general, there are two types of IoT data observer models in healthcare:

Model one – IoT devices from the same vendor attached to a single patient collect different datasets from the patient, route them through a single aggregator and then distribute this data to vendors. In this model, multiple observers and collectors are there with a single data plane[i].

Model two – IoT devices from different vendors attached to a single patient collect different datasets from the patient in different frequencies. In this scenario, there are multiple collectors collecting data and sharing it with different observers in different data planes[i]. Aggregation of data collected from multiple IoT devices is done by data application or healthcare professionals, for an understanding of a patient's overall health.

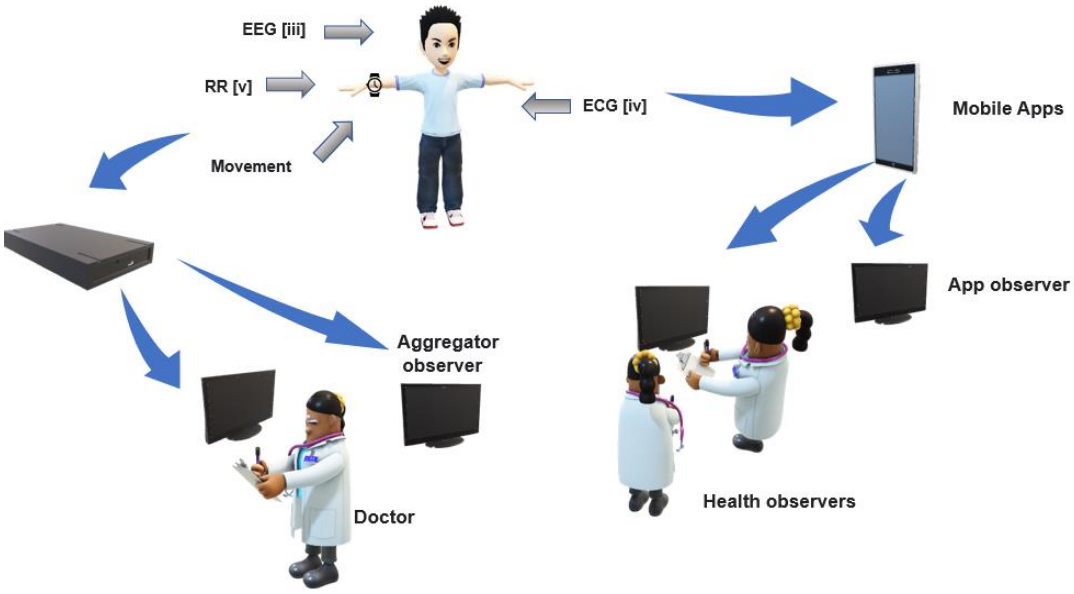

Figure 1: Fitness devices and observers

1.2 AVIATION

Aeronautical telemetry collection involves telemetry collected from different components of an aircraft and shared with different observers. Though telemetry collection from aircraft started with multiple sources and a single data plane, with the increasing number of sources and varied datasets, the topology of telemetry transformed with multiple observers needing different datasets and hence data planes. In aircraft flight testing, multiple telemetry channels with distinct receiving sites are required for covering blind spots as well as to provide an additional data source with independent noise.

Figure 2 (courtesy: Adex Aerospace)

1.3 SERVER INDUSTRY

In Servers, telemetry data is used to improve customer experience, monitor security, system and application health, quality, and performance. With the growing importance of data analytics and its impact on customer experience, telemetry has become an inevitable part of servers. SNMP has been using polling technology to collect system-related data for years. Recently, pushing and streaming [ii] of data has gotten more attention as there is an increase in the need for collecting several types of telemetry data from distinct sources for distinct observers.

A server is made up of multiple subsystems such as storage, CPU (Central Processing Units), filesystems, etc. Each subsystem has its telemetry collected and sent to its respective manufacturers. The same data or subset of data is used by other monitoring applications across other subsystems. Here, there are multiple observers: the manufacturer, system monitoring application, a data collector for analytics, and so on. Each observer might collect the same or different data. Figure 3 depicts a server telemetry setup with multiple observers.

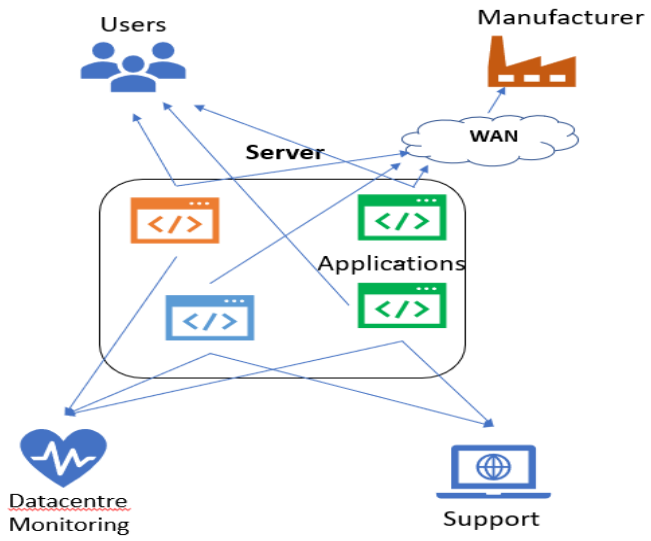

Figure 3: Server telemetry topology

2. CHALLENGES WITH MULTIPLE OBSERVERS

Many challenges are faced by telemetry setups with multiple observers and collectors.

The first issue is known as the **observer effect** and this is exacerbated with multiple observers. The Observer effect is the disturbance of an observed system by the act of observation. In servers with multiple observers and telemetry collectors, the act of collection can negatively impact the performance of the system. Telemetry sources will need additional system resources like CPU, memory, and storage for processing and storing required telemetry data. In addition to this, telemetry collectors can consume network bandwidth while sharing data with observers. These can impact the performance of the core functionalities of a server.

The second issue is related to the **security** of the system. With multiple observers at various locations, there are more paths where data is sent and hence security aspects of dealing with multiple networks must be considered. Sharing telemetry data using insecure methods with multiple observers can expose system details compromising the system security.

The third and major area where observers can impact a system under observation is its **compliance with regulations** like GDPR (The General Data Protection Regulation). With different observers, each observer could be in a different location thus the challenge is to know each observer's location and comply with the regulations of the respective location.

The fourth issue is about **data inconsistency** that pops up because every application is collecting and sending data from different sources and in varied frequencies. This can lead to multiple observers getting inconsistent data.

As the number of observers increase, so does the challenge of meeting these requirements while minimizing overall system impact.

3. PROPOSED ARCHITECTURE

To address the majority of the mentioned challenges, in this section, we are proposing a design that is generally applicable to telemetry systems. Section 3.1 explains how to categorize telemetry data and Section 3.2 explains the architecture in detail.

3.1 TELEMETRY DATA CATEGORIZATION

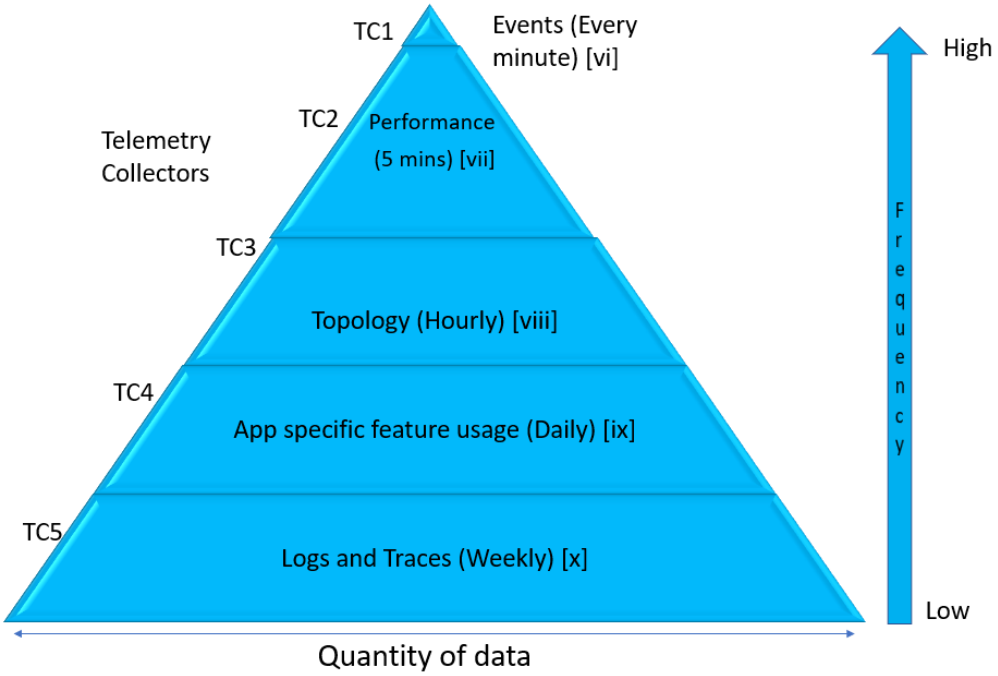

Figure 4: Telemetry collectors and categories

In the proposed design, telemetry data is grouped into various categories and each category is mapped to a frequency. The pyramid in figure 4 shows 5 categories though the number of categories depends upon the specific system under observation and types of data needed by its observers. For every category, there is one telemetry collector process that runs and collects its respective set of data in the pre-defined frequency. The frequency for a specific category is specified in brackets in figure 4. For events/alerts, the respective Telemetry Collector (TC1) is designed to run frequently (say every 1 minute). Similarly, TC2 designed to collect performance-related datasets is tuned to run every 5 mins and so on. In summary, the pyramid in figure 4 depicts categories of telemetry data mapped to a specific telemetry collector which will run and collect at a specified frequency. Now let us take a look at the architecture of the proposed solution.

3.2 TELEMETRY ARCHITECTURE

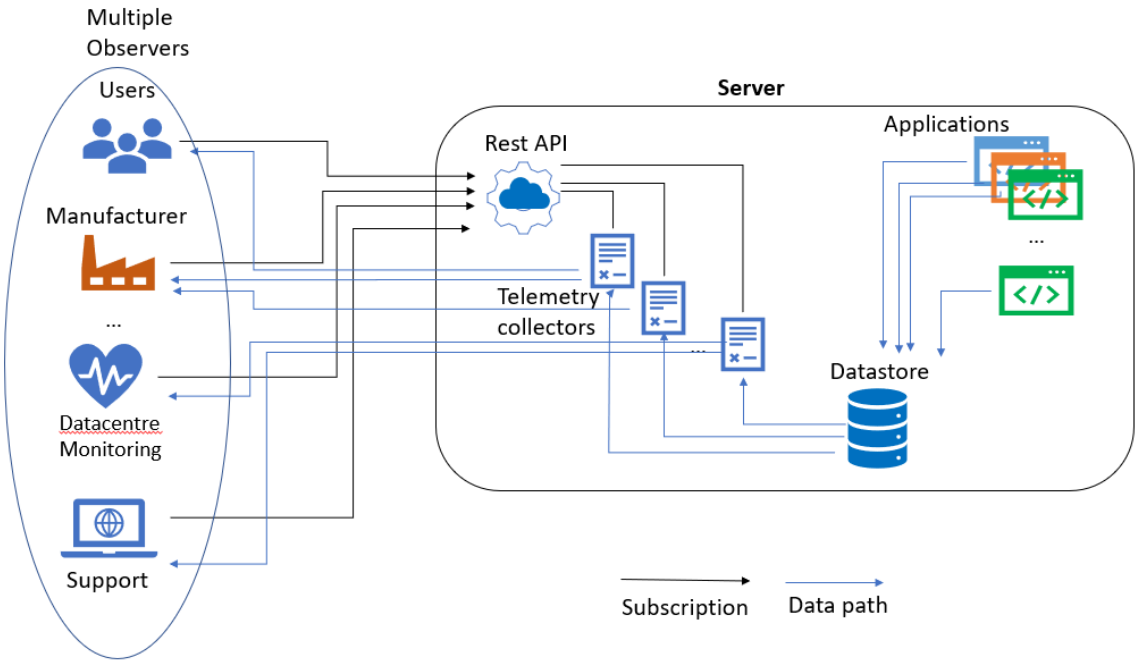

Figure 4: Proposed telemetry architecture

Applications are provided with a data store into which they can store data or pointers to data (in cases of logs or trace files). These applications push data into the data store.

Specific telemetry collectors run at pre-defined frequencies to collect the allotted data set. For example, TC4 will run every 5 minutes to collect system performance data.

Observers are provided with REST interfaces to subscribe to one or more telemetry data categories. Once subscribed to telemetry data categories, a REST interface will attach those observers to the respective telemetry collectors along with the protocol and destination information. A telemetry collector will then distribute the same data set to multiple subscribed observers.

In this design, multi-layer data planes are used when telemetry collectors share data with observers using different types of network connectivity. From one observer's view, the data traffic routed to this observer by one or multiple telemetry collectors constitutes a single data plane. As there are multiple observers and multiple collectors, there are multiple data planes involved.

Pros:

1) Every type of dataset is read, only once from the data store by one of the telemetry collectors. This reduces the “observer effect” significantly though multiple observers are collecting data from the same system.
2) Pushing and streaming technology is used than pull technology by default, so that data collection is continuous and predictable. In the enhanced design, observers can pull data on-demand through pre-defined REST interfaces.
3) A REST-based subscription gives the flexibility to add a new observer or remove an existing observer without any change in the architecture.
4) Applications that are data sources are decoupled from the data collection flow.

Cons:

1) Telemetry data needs to be categorized and collectors implemented for these categories.

4. COMPARISON WITH EXISTING ARCHITECTURES

In Sections 1 and 2, we presented several types of telemetry architectures where multiple observers collect data from different data sources. Figure 3 shows a server architecture in which individual applications try to share data with respective observers. All these architectures face the challenges that we discussed in Section 2.

We can have a closer look at the proposed architecture in section 3 to find out how it addresses these challenges faced by other architectures.

1) In the proposed design, observers are fed with required data in a controlled manner. The number of data collectors is limited. A data collector can share the same type of data with multiple observers. This optimizes data collection as well as resources like CPU, memory, storage, and network bandwidth usage by data collectors in the system. Application data is routed to a data store to avoid every application trying to send data outside the system and causing network bandwidth issues and moves network security aspects, w.r.t observers, away from apps to collectors. The current design gives more control to the system over the data collection and hence reduces the **observer effect**.
2) In cases, where every application sends its data externally to a server, the server will not be able to provide enough security checks to see whether the data transfer is secure per application. In the proposed design, observers subscribe through REST interfaces and the system can put in required validation to validate the connectivity details and protocols registered by the observer are secure and adhere to the appropriate regulations. Observers are not allowed to collect data directly from source applications. Instead, pre-defined telemetry collectors handle data traffic through streaming or pull technologies which ensure the server can apply best **security** practices and remove the burden from applications.

3) **Compliance with regulations** will be difficult in architectures where every application is sending its data to different observers and the server does not have control over that. In the proposed architecture, the number of collectors and the category of data that every collector shares are pre-defined. Telemetry collectors sharing application-specific data can identify the observer's location and filter/modify the data accordingly to comply with regulations based on the observer's location.

4) In addition to addressing multiple major challenges, the proposed design helps in providing a flexible interface through REST for new observers to subscribe. An additional advantage is that when observers register through a REST interface, the server can apply required throttling policies so that data collection does not impact the performance of the system.

5) The architecture allows more observers and data types to be added dynamically and the data collection frequency to be modified based on requirements. This can be extended further to spin down a collector in case no observer has subscribed to it. Further optimizations and enhancements are feasible based on the environment.

6) Telemetry collectors collect specific type of data from data store and share copies of the same dataset with multiple observers. This ensure that shared data is consistent across all subscribers.

CONCLUSION

When we observe any industry (Aerospace, Transport, Computer, Telecommunication, etc..), we can see that importance and impact of data analytics is increasing day by day. This automatically brings telemetry into the spotlight. With every application owner, device owner, and user looking for inferences from data collected through telemetry, the scenario of multiple observers for a single system is going to be common in near future. Here, we proposed a generic design for servers that is adaptable across the industry. This proposal will help the community to focus on this scenario and trigger discussions. We are hoping this will lead to more optimized designs in the future in various areas like IoT, Edge devices, Servers, etc.

GLOSSARY

[i] **Data plane**: Software that processes data traffic by copying bits and bytes from the source of the traffic and sends them to its destination.

[ii] **Pushing and streaming**: Instead of data getting pulled by the observer, data gets pushed by the system under observation. When pushing of data is done for any state changes of the data source, then it becomes streaming of data.

[iii] **EEG**: An electroencephalogram (EEG) is a test that detects electrical activity in the brain using small, metal discs (electrodes) attached to the scalp

[iv] **ECG**: An electrocardiogram (ECG) a test that records the electrical activity of the heart through small electrode patches.
[v] **RR**: The respiratory rate (RR)is the rate at which breathing occurs, usually measured in breaths per minute
[vi] **Events**: Alerts generated for specific events like 100% space usage, disk error, etc.
[vii] **Performance**: Values related to resource usage in the system like CPU usage, Memory consumption, Network bandwidth usage, etc.
[viii] **Topology**: Configuration of a system like number of disks, number of network cards, etc.
[ix] **App-specific feature usage**: Usage information related to specific features of applications like how many times the user triggered a specific feature.
[x**] Logs and Traces**: Log and Trace files captured for specific applications or overall system.

REFERENCES

1. Data plane, control plane, and their APIs explained: https://medium.com/@aburnos/data-plane-control-plane-and-their-apis-explained-d0a3fa7291f3
2. Telemetry Patterns in IoT: https://iotatlas.net/en/patterns/telemetry/
3. The next decade is IoT's decade: https://www.avenga.com/magazine/wearables-iot-healthcare/
4. Aeronautical Telemetry: https://www.curtisswrightds.com/applications/aerospace/flight-test/challenges-in-telemetering-data.html
5. Multihop Airborne Telemetry networks: https://www.researchgate.net/figure/Aeronautical-Telemetry-Network_fig1_228573274
6. Arch for streaming data related to networks: https://www.kentik.com/blog/how-to-maximize-the-value-of-streaming-telemetry-for-network-monitoring-and/
7. Packet level telemetry in large DCNs: https://conferences.sigcomm.org/sigcomm/2015/pdf/papers/p479.pdf
8. Link on GDPR: https://www.csoonline.com/article/3202771/general-data-protection-regulation-gdpr-requirements-deadlines-and-facts.html
9. Software Telemetry styles: https://livebook.manning.com/book/software-telemetry/chapter-1/v-5