FABRIC Measurement Framework Design

v0.2

1. Introduction **2**

2. Components of Measurement Framework **3**

3. Relationship of the Control and Measurement Frameworks **7**

4. Measuring the FABRIC Infrastructure **8**
4.1 Measurement Points 9
4.2 GPS-disciplined Timestamps 9
4.3 Measurement Bus 9
4.4 Data Collection, Filterning, and Storage 10
4.5 Web-based GUI Operator Tools 10
4.6 Setting up Services Used in MF 10

5. Instrumenting and Measuring FABRIC Experiments **11**
5.1 Types of Experimenter Measurements 12
5.2 Provisioning Experiment-Specific Resources 13
5.2.1 Provisioning VMs/Containers to perform ELK/Prometheus functionality 13
5.2.2 Provisioning QoS capable Experiment-specific Measurement Busses 14
5.2.3 Provisioning for Packet Capture 14
5.3 Timestamping Services 14
5.4 Experiment-Specific MF components 16
5.5 Combining Experiment-specific measurements with Infrastructure Measurements 17
5.6 Storing Experiment-specific Measurement Data 17
5.7 Visualizing Experiment-specific Measurement Data 17

6. Security: Authentication, Authorization, Privacy **18**

7. Measurement Information Model **18**

8. MF API **19**

9. MF and the Message Bus **19**

Document Version	Date	Description
v0.1	03/24/2020	Initial version
v0.2	11/26/2020	Update the content for instrumenting and measuring FABRIC experiments, revised for release before the Facility Partners Workshop
v0.3	09/01/2021	Updated to reflect integration with the CF information model and relationship with the Centralized Data Store clusters.

1. Introduction

The **Measurement Framework (MF)** is designed to assist both ***operators*** who need to monitor the FABRIC infrastructure and ***experimenters*** who want to measure the components of their experiment (and underlying infrastructure). The Measurement Framework assists with tasks such as instrumenting the components of an experiment, measuring a component, collecting and storing measurements, filtering and processing measurements, and visualizing measurements. It is capable of collecting a wide range of "measurements" including numeric measurements, event/log entries, and packet traces.

The Measurement Framework supports measurement of the *FABRIC infrastructure* (e.g., the FABRIC hardware including compute, storage, switches, PDUs, and various FABRIC services like the AM and Control Framework services) **and also *experimenter resources*** including virtualized resources (e.g., VMs and their guest OS) and pass-through hardware (GPUs, FPGAs, NICs) . **The Measurement Framework also includes a set of *MF services*** that collect and manage measurement data including services to store measurement data, filter, process, and search measurement data, and display/visualize measurement data. The MF will provide programmatic ways (API calls) to enable/disable measurements and access measurement data. It will also support web-based GUIs to interactively display and work with measurement data. While the MF is able to capture and store experimenter's measurement

data, there are limits to FABRIC's storage capabilities. Long term archival storage of measurement data may require moving data to external (non-FABRIC) storage resources. While the task of moving data to long term archival storage will initially fall on the experimenter, the desire is that the MF will eventually provide high-speed data transfer capabilities to frequently-used storage resources (e.g., commercial cloud storage, HPC/DTN storage, or Open Storage Network (OSN) resources).

The MF offers several salient features not found in other network measurement systems. For example, the transmission of measurement data can be designed such that it does not interfere with an experiment's data plane traffic, in some cases being transferred over the Internet and in other cases using a dedicated "measurement channel" across ESnet with isolated performance properties and possibly QoS guarantees. Another feature of the MF is the ability to timestamp events and measurements with a high-precision GPS-synchronized timestamp (a.k.a., *Packet GPS*) that results in precise and comparable timestamps across the FABRIC network to the tens of microseconds. The MF is also capable of line-rate full packet capture using port mirroring and dedicated NIC cards to capture and timestamp packets. (Line-rate packet capture may be limited in duration by the amount of space available to store captured packets). In addition, ESnet's high-touch services may be available to filter, process, and archive packets at line-rate using FABRIC's programmable NIC/FPGA cards. Other (hidden) features such as the MF being integrated into the Control Framework ensure that MF resources are explicitly allocated and accounted for, which aids in understanding the cost of doing measurements and can be used to inform resource allocation decisions.

The Measurement Framework leverages the Control Framework to allocate measurement resources on behalf of the experimenter. After measurement resources have been allocated, the Measurement Framework configures and enables the measurement resources and measurement services to begin collecting measurement data. Measurement data may be collected and stored locally at the resource, or may be sent to MF *storage, processing, and visualization services* that can be accessed by experimenters over the FABRIC IPv6 dataplane.

2. Components of Measurement Framework

The Measurement Framework consists of six components (see Figure 1):

- **Measurement Instrumentation and Control Service** -- that instruments user experiments with measurement capabilities and can dynamically enable/disable measurements.
- **Measurement Points (MPs)** -- that collect data about the running experiment, including state information and events (user-defined events, packet arrival/departure events, etc).
- **Timestamping Services** -- that MPs can invoke to accurately timestamp events (using globally synchronized clocks with tens of microsecond accuracy or better).

- **Measurement Data Storage Services** -- that either store collected data locally on a rack, somewhere within an experiment or project, or globally in a FABRIC shared storage space.
- **Measurement Data Filtering/Processing Services** -- that filter or transform the data into a format that can be easily analyzed and visualized.
- **Experimenter Measurement Tools** -- that enable experimenters to control their measurements and access/search/analyze/visualize their measurement data.
- **Measurement Bus** -- a conceptual bus (implemented in multiple ways) that carries measurement data between the various MF components.

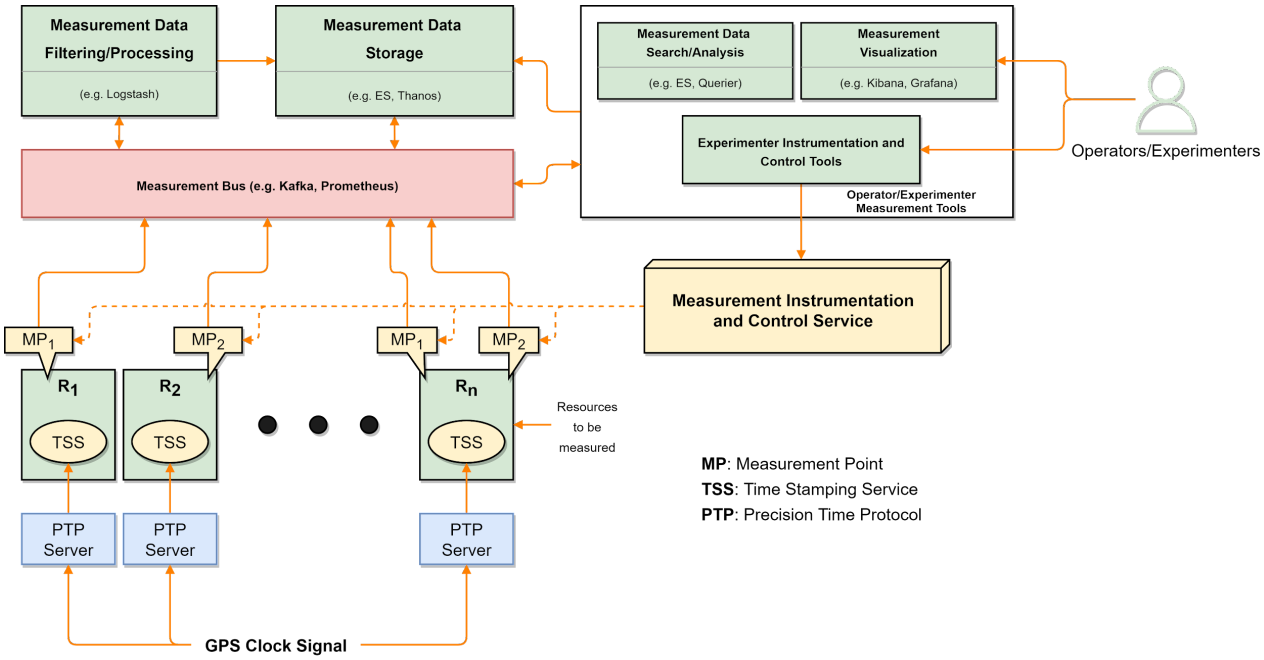


Figure 1: The MF components

The Measurement Framework supports measurement of the FABRIC infrastructure (e.g., the FABRIC hanks) as well as the experiments that run on the infrastructure. FABRIC Infrastructure measurements are used by operators (with some data being publicly available to experimenters), whereas experiment measurements are typically used by the experimenter who owns the experiment. In other words, the MF components described above that measure the (relatively) static FABRIC infrastructure are largely provisioned and set up as part of the FABRIC infrastructure deployment. They are used by operators and experimenters alike -- although experimenters are blocked from accessing certain privileged measurement information. On the other hand, MF components used to measure an experiment are experiment-specific and must be dynamically provisioned and set up for each experiment. They are used by the experimenters that own the experiment.

The goal of the **Measurement Instrumentation and Control Service** is to dynamically provision and deploy the MF components described above and allow experimenters/operators to control the deployed components. In the context of the FABRIC infrastructure, the MF components are largely deployed along with the infrastructure, but certain capabilities -- such as the desire of an operator to capture packets at full line speed for monitoring or debugging purposes -- may require dynamic deployment and provisioning of dedicated infrastructure measurement resources. In the context of an experiment, the Measurement Instrumentation and Control service must deploy and provision the MF components along with the experiment components. The Control Framework is responsible for provisioning both experiment resources and measurement resources. Consequently, the Measurement Instrumentation and Control Service is implemented as a service that leverages the Control Framework and is responsible for embedding, provisioning, configuring, enabling, and controlling measurement resources on behalf of experimenters (or system operators).

Although the MF Instrumentation and Control Service calls the CF to allocate resources, the MF and the CF are tightly integrated in the sense that the MF is able to augment the resource information model maintained by the CF. Measurement resources allocated by the MF on behalf of the experimenter are simply resources from the CF's perspective. In other words, the CF is unaware that the allocated resources are being used for measurement purposes. However, the CF allows the MF to augment any of the objects in the CF information model (elements of an experimenter's slice) with information about how they are being used by the MF. For example, consider a VM in a slice that is represented by an object in the CF information model. When the MF enables (runs) a high-precision GPS clock synchronization service in the VM for the purpose of timestamping, the MF will add a tag to the object in the CF information model indicating that the VM has a GPS-synchronized clock. Or consider a VM that will be used to store measurement data from the experiment, and the MF sets up a Prometheus database and service on the VM to collect data from other VMs in the slice. In that case, the MF will augment the corresponding object in the CF Information model with information about the Prometheus service including the IP address where the user can access the Prometheus web interface. The CF ignores these extra pieces of information, but by allowing the MF to add extra information, the MF is able to utilize the same information model being used by the CF.

The MF differentiates between *Provisionable Measurement Resources (PMRs)* and *Accessible Measurement Resources (AMRs)*. Provisionable Measurement Resources refer to a resource that the MF Instrumentation Service allocates, provisions, and configures such as a MF VM attached to a mirror port for the purposes of packet capture, or a MF VM provisioned to store measurement data, or a MF VM provisioned to filter measurement data. Accessible Measurement Resources are measurement capabilities that are part of experiment resources and simply need to be enabled and accessed. Examples include an experiment VM that contains an SNMP daemon that is just part of the VM and can be directly accessed by the user, or the CPU/Memory/Disk load averages caused by an experiment VM (maintained by the virtualization system), or the power-meter associated with a network switch. All of these AMRs

are simply part of an experimental resource. The Measurement Instrumentation Service is only responsible for provisioning PMRs. AMRs are typically controlled by MF Tools directly, or possibly through the Instrumentation Services (pass through) invoking a POA (Perform Operation Action) plugin in the CF AM.

A **Measurement Point (MP)** may be either a PMR or AMR resource. For example, an experimenter may ask for an MP to capture the packets arriving at a port on the data switch, resulting in the creation of a MF VM connected to a mirror port on the switch (i.e., a PMR). Alternatively, an experimenter may ask for an MP that captures packets arriving on an ethernet interface (e.g., eth0) on a particular experiment VM, resulting in tcpdump being invoked on the interface (i.e., an AMR). Experimenters can also create their own experiment-specific measurement points. In short, measurement points measure some aspect of the experiment or the underlying FABRIC infrastructure. Measurement points are typically located on the resources themselves (see "Measurement Capabilities" in Figure 2).

A key feature of FABRIC is the ability to timestamp events with a high precision timestamp (tens of nanosecond resolution) that is accurately synchronized (tens of microseconds) across all FABRIC nodes. To achieve this, the measurement framework supports a **Timestamping Service** based on PTP and GPS timing signals. Measurement points can use this service to timestamp measurement events accurately and then compare the time that events occur across FABRIC nodes. Consequently, the measurement framework is designed around the concept of time-stamped measurement events and their associated data. Timestamping services are typically located on the resources themselves (see "TSS" in Figure 1).

MF **Data Storage Services** may be local to a node (worker node or VM), local to a FABRIC rack (on the same rack), or may be on a remote FABRIC rack collecting experiment data (globally) from many FABRIC nodes. Local data collection may be necessary for collecting large amounts of measurement data at high speed -- say to perform packet capture at tens of gigabits per second. Global data collection is useful for capturing events and periodic measurements/metrics across an experiment.

Pipelines of **Measurement Data Filtering and Processing services** can be constructed within an experiment to filter, transform, analyze, and search measurement data from MPs in the experiment. Like the Data Storage Services, the placement of the Data Filtering and Processing services can be on a worker node, FABRIC head node, or some remote location. Placement of such services is handled by the MF Instrumentation Service.

Note that the pipeline of MF Processing, Storage, and Visualization services transmits data between services via a conceptual MF **Measurement Bus** (see Figure 1). The Measurement Bus will typically be dynamically created as part of the Instrumentation process, tailoring the implementation of the Measurement Bus to the measurements being collected. In most cases, the requirements on the Measurement Bus will be minimal, but in some cases, the Instrumentation service may need to allocate FABRIC links with QoS to meet the needs of the

measurement data collection. Because CF resources, including measurement resources can be allocated with QoS, one can view the Measurement Bus as a measurement plane, distinct from the data plane, having its own dedicated bandwidth.

Distinct from, and in addition to the Measurement Bus, all MF components have access to the CF Message Bus for communicating and logging actions/requests between components.

3. Relationship of the Control and Measurement Frameworks

The relationship between the *Control Framework (CF)* and the Measurement Framework (MF) is illustrated in Figure 2. The Control Framework is responsible for provisioning experiment and measurement resources, while the *Measurement Framework* is responsible for operating the measurement resources, assisting the experimenter in collecting measurement, processing, managing, storing and visualising the measurement data. CF and MF work in concert to determine and provision the combination of resources necessary to conduct the experiment and measure its outcomes.

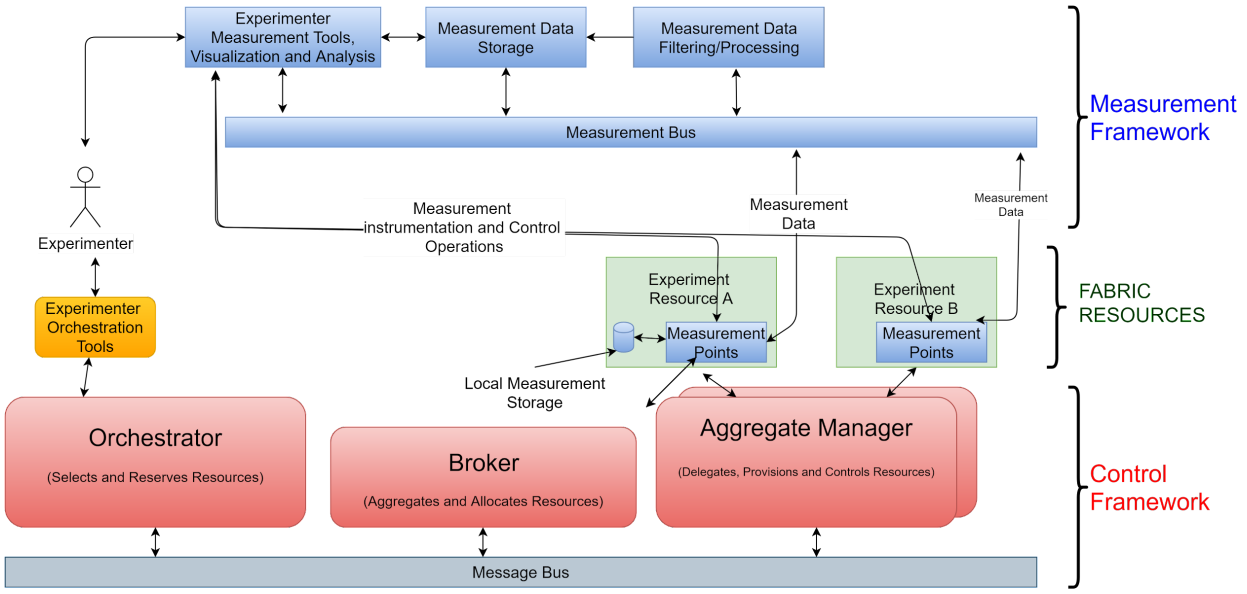


Figure 2: The Relationship between the CF and MF

As noted earlier, the Measurement Framework leverages the Control Framework to allocate measurement resources needed by an experiment. After measurement resources have been allocated, the Measurement Framework configures and enables the measurement resources and measurement services to begin collecting measurement data. Measurement data may be collected and stored locally at the resource, or may be sent to MF *storage, processing, and visualization services* that can be accessed by experimenters over the FABRIC IPv6 dataplane.

4. Measuring the FABRIC Infrastructure

To measure the FABRIC infrastructure itself, MF components are deployed across all FABRIC racks to collect data and send it to centralized data storage services where it can be analyzed and viewed by operators. Figure 3 below illustrates the MF services -- a.k.a., Measurement Points (MP) -- deployed on the racks and the flow of data from the MPs to the centralized MF data storage services. Numeric measurements are collected using Prometheus exporters running on the head and worker nodes, are cached in a Prometheus DB on the rack, and eventually sent to a centralized data store (Thanos) where they can be viewed by operators via Grafana. Event logs and packet information is collected with ELK beats running on the head and worker nodes that are sent via a centralized Kafka service to an ELK DB that can be viewed by operators via Kibana.

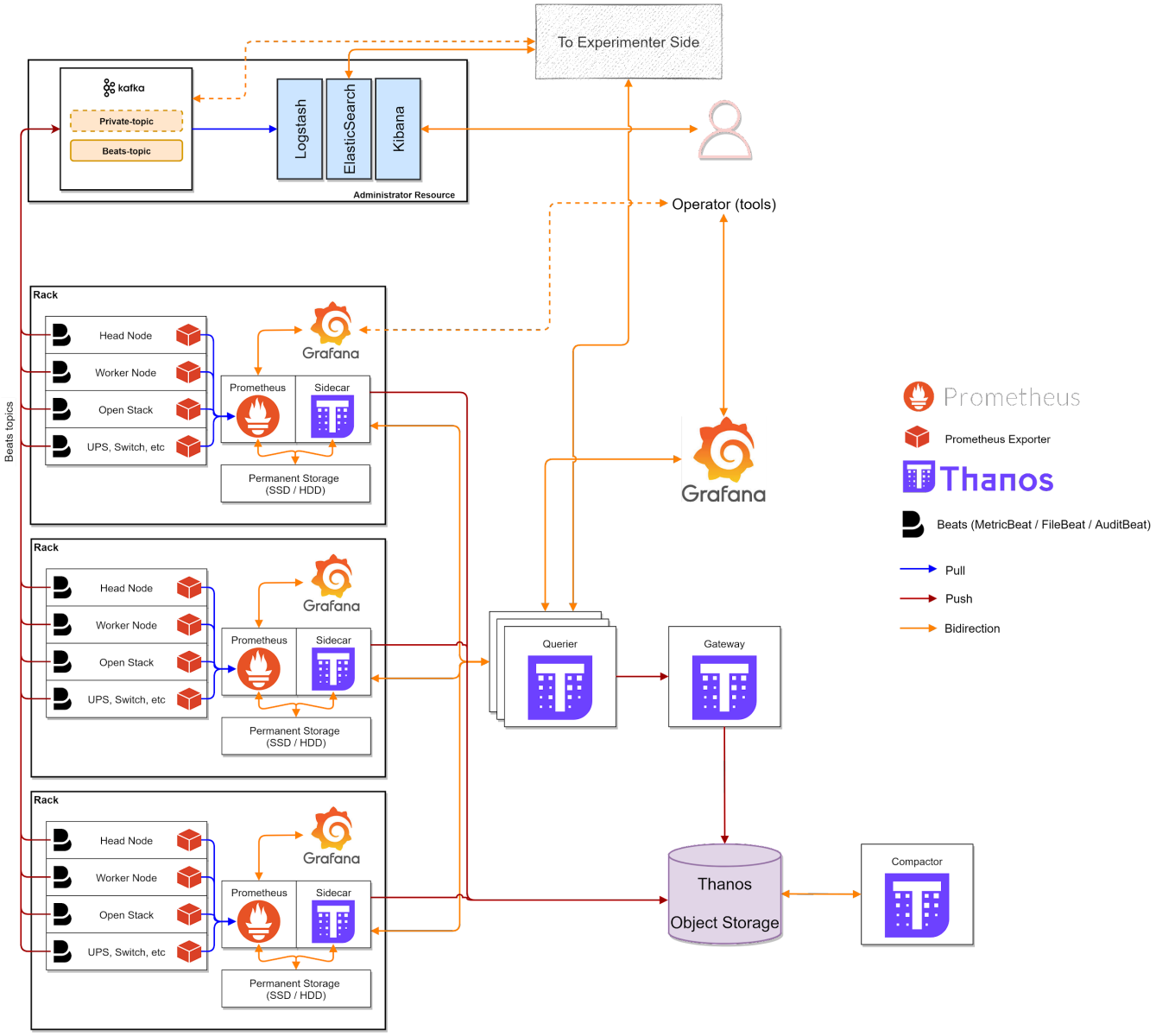


Figure 3: MF components that collect measurements from the *FABRIC Infrastructure*

As noted earlier, these MF Components are largely provisioned and deployed as part of the FABRIC infrastructure deployment. The MF Components used to measure the FABRIC Infrastructure consist of measurement points on FABRIC head/worker/switch devices, GPS-disciplined clocks used for timestamping, pub/sub “measurement bus” services, data collection/filtering/processing/storage services, and web-based GUI tools to explore the measurement data.

4.1 Measurement Points

A variety of measurement points are initially configured to continuously measure (for monitoring purposes) the general health of the FABRIC Infrastructure. General health examples include CPU, memory, disk, and network bandwidth usage collected from the Head and Worker nodes of every FABRIC rack. The actual list of general health metrics collected continuously from each FABRIC rack include a relatively long list of values (see Measurement Matrix). Some of these measurement points are implemented with Prometheus exporters collecting and sending data to a Prometheus server running on the Head Node. Other measurement points are implemented using ELK beats. For example, ELK filebeats are used to collect log entries from various system log files continuously monitoring events on the Head and Worker nodes. These events are particularly useful for identifying errors and failures that occur.

4.2 GPS-disciplined Timestamps

Each FABRIC rack has a GPS-disciplined PTP server that pushes out GPS-synchronized clock signals to the management NIC in the Head and Worker nodes. This signal is used to set the host operating system clock. The host operation system clock is used to timestamp events generated by the Prometheus and ELK beat measurement points.

4.3 Measurement Bus

There are effectively two measurement buses: a pub/sub bus for ELK-based measurement points, and Prometheus/Side-car data buffering and stream service for Prometheus-based measurement points.

The first measurement bus uses the Apache Kafka distributed pub/sub messaging system. Beats (e.g., filebeat, packetbeat) are installed on nodes in FABRIC racks as Measurement Points that collect event/log entries, such as arriving packet headers or system log messages. They are the producers that publish the collected information tagged by topics to the pub/sub measurement bus. The topics can be used to determine what type of measurement data is being sent and may also be used to determine whether the data is public or private. The MF data filtering/processing services typically subscribe to all topics, while individual experiments are only allowed to subscribe to topics they are allowed to access.

The other measurement bus -- for data collected by Prometheus exporters -- is designed to efficiently handle numeric metric data, such as CPU load, memory usage. This bus is implemented using a Prometheus server and ThanOS side-car installed on the Fabric head node. It transports data by pulling data from exporters running on all nodes in the rack and then buffering the data locally before forwarding the data to the combined rack measurement object store that can be searched by the Thanos Querier.

4.4 Data Collection, Filterning, and Storage

Event/Log entry measurement data are processed by, and stored in, the MF ELK service (a replicated service for reliability purposes that is used by all FABRIC racks), consisting of Logstash, Elastic search and Kibana. The ELK system subscribes to the measurement bus (Kafka) topics and collects the data to be processed. First, the data is handled by Logstash to derive the structure and key fields, filtering and transforming the data as needed to be indexed by the Elastic search database. Second, Elastic search indexes and stores the data for future access by operators. Operators can then issue Elastic search queries to extract and analyze data of interest. Finally, the data can be visualized by using Kibana for presentation to operators via dashboards consisting of charts and graphs highlighting information of importance.

Numeric measurements are sent by the prometheus/side-car steaming service to the MF ThanOS Object Store which maintains a history of performance data going back in time. Note measurement data is buffered at each FABRIC rack and is not streamed immediately to the MF's ThanOS Object Store. In order for operators to see the most up-to-date information, the data must be requested from each of the FABRIC racks. When ThanOS Querier receives a request for data, it obtains historical data from the MF ThanOS Object Store but then requests the most recent data from Sidecar running on each of the racks.

4.5 Web-based GUI Operator Tools

Two GUI tools are used by the MF for visualizing the measurement data. One is Kibana from the ELK-system. It can categorize measurement data and present them at different abstract levels. Grafana visualizes the streamed metric data from Prometheus. Both GUI tools let operators define customized dashboards and focus on the measurement data of interest. For example, operators can visualize how the specific metric changes over a selected period of time.

4.6 Setting up Services Used in MF

MF depends on running several services for collecting/filtering/processing/presenting measurement data. Detailed setup instructions and related scripts are described in the MeasurementFramework repository on GitHub.

5. Instrumenting and Measuring FABRIC Experiments

In addition to measuring the underlying FABRIC infrastructure, the Measurement Framework (MF) is also designed to help users measure their experiments. In particular, the MF assists users in instrumenting their experiments with measurement capabilities after the experiment has been provisioned. Because each experiment has a different topology and resource configuration, the MF instrumentation services must tailor the deployed measurement

capabilities to fit the experiment. In addition, the MF offers measurement point provisioning services that dynamically allocate and configure the resources needed to implement provisionable measurement point resources. Having provisioned and instrumented an experiment with measurement services, the MF provides users with the ability to dynamically control the measurements in their experiment via API-based interfaces associated with the Instrumentation and Control Service. The MF is also capable of instrumenting an experiment with visualization capabilities that users can interface with to view and explore the measurements coming out of their experiment. It should be noted that the experiment-specific measurement components are the same as the infrastructure measurement components described earlier, so we will not re-describe each of these components, but rather simply mention that it is the instrumentation services responsibility to deploy the same set of components within an experiment as are used to measure the underlying infrastructure.

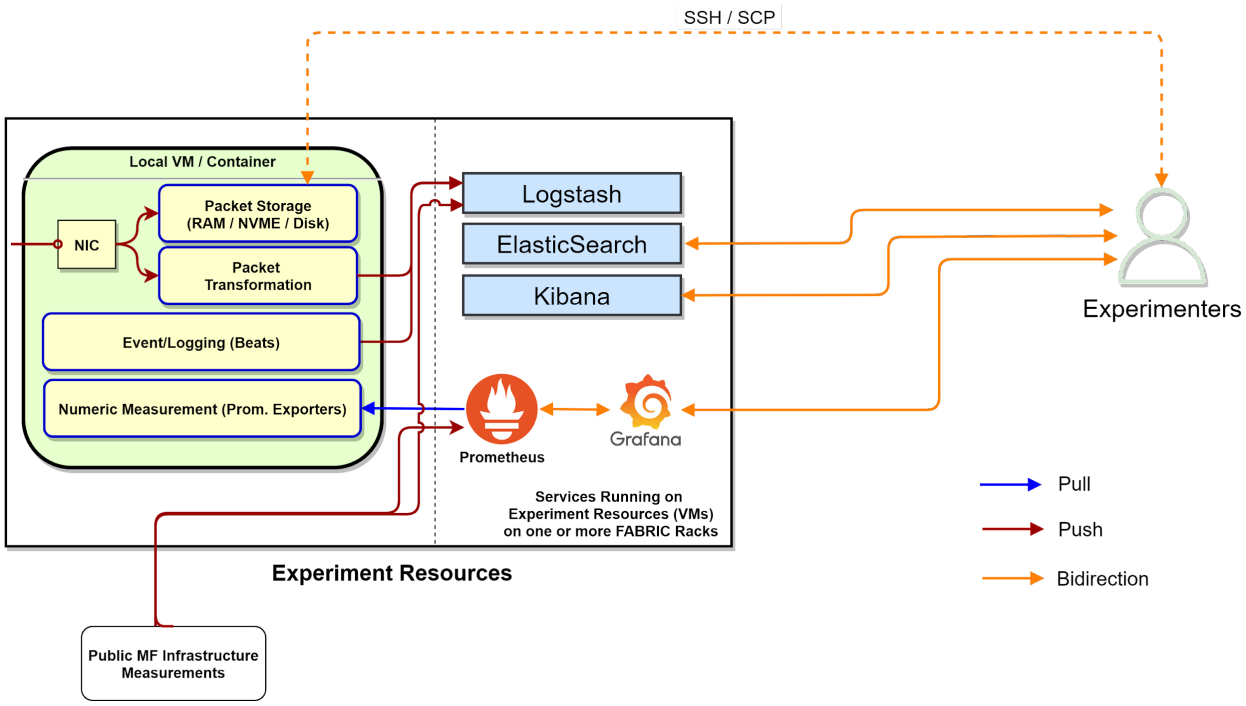


Figure 4: Experiment-specific Measurement Services

Figure 4 illustrates the Experiment-specific measurement services that can be dynamically instantiated and deployed within a provisioned experiment. In particular, the MF can automatically deploy an ELK or Prometheus system within a user's experiment on their behalf, allowing experimenters to capture and search, analyze, and visualize the same types of measurements that are being measured by the underlying FABRIC infrastructure, but in this case, capturing measurement information only from within the experiment. In addition, the MF can instrument an experiment with packet capture capabilities, that either capture (and store locally) full packets or capture (and process/store) packet headers. In addition, the MF is able to read measurement data from the underlying FABRIC Infrastructure and combine it with experiment-specific measurement data to form a more complete understanding of what is going

on in both the experiment and infrastructure at any given point in time. Lastly, the MF is capable of deploying GUIs and search services that experimenters can use to access the measurement data for their experiment.

5.1 Types of Experimenter Measurements

Measurements support by experiment-specific MFs roughly fall into one of three categories:

- Numeric measurements
- Event/Log messages
- Packet capture

Like the infrastructure measurement systems, users can collect and explore standard numeric performance information (e.g., cpu load, memory usage, disk utilization, etc from experiment resources such as VMs) as well as event/log messages (e.g., system logs from experiment guest OSes). An additional measurement frequently needed by experiments is the ability to capture packets to understand the network traffic.

While the Infrastructure MF focused on industry standard metrics obtained via Prometheus exporters and various ELK beats, experiment-specific MF capabilities allow experimenters to collect their own experiment-specific numeric measurements (e.g., the number of items currently in an NDN cache) of the log messages output by any experiment specific service (e.g., the NDN forwarding service). Similarly, experimenters can capture experiment-specific information from packets transmitted over the experiment topology.

5.2 Provisioning Experiment-Specific Resources

While the infrastructure measurement service runs on pre-allocated resources in the Fabric infrastructure or Fabric Centralized Data Services, the experiment-specific measurement service may need to run on a separate VM or container, or use a dedicated NIC and links. These resources must be dynamically allocated and provisioned on Fabric racks, and accounted for and charged to the experiment using them.

The MF Instrumentation and Control service is responsible for provisioning the measurement resources requested by experimenters. It does so by invoking the CF to allocate and provision the necessary resources on the FABRIC rack to create the desired Measurement Point (MP), and then configuring and enabling the MP. In addition to provisioning MPs, the MF Instrumentation and Control Service may also need to configure experiment-specific ancillary services needed to collect, filter, search, and visualize the experiment-specific measurement data.

5.2.1 Provisioning VMs/Containers to perform ELK/Prometheus functionality

Similar to the ELK/Prometheus functionalities provided to the infrastructure measurement, the MF can equip an experiment with its own ELK/Prometheus functions to collect, filter, and visualize the data collected for the experiment. To run these services, the MF allocates VMs/Containers as measurement points via the CF. These resources will be considered as a part of the slice for the experiment.

Determining where in the slice these services should be deployed is the job of the Instrumentation and Control Service. In most cases they will be deployed in a hierarchical fashion with measurements initially aggregated locally in an experiment VM on a FABRIC rack and then being sent to a central collection point within the slice (say to a slice VM running on a centrally located FABRIC rack -- e.g., on the FABRIC Centralized Data Storage Cluster). Users may be able to specify how they want their experimenter-specific ELK/Prometheus services deployed within the slice, but are expected to typically rely on the Instrumentation Service to make sensible decisions, freeing users to focus on the design of their experiments.

5.2.2 Provisioning QoS capable Experiment-specific Measurement Busses

Experiment-specific measurement data can be stored locally in the VMs they are collected at or on VMs local to the FABRIC rack on a temporary basis, but the resources at a FABRIC rack are limited, meaning that longer-term storage of measurement data will require moving it elsewhere. Moreover, users may want to aggregate data coming from multiple racks in one place which will also require moving measurement data off-rack to some remote location. Experimenters may want to ship the data to a remote location either live when the experiment is ongoing or after the experiment has finished.

To avoid interfering with the data plane of the experiment, the MF can set up a QoS capable channel as a measurement bus for transferring the data to a remote location, such as google drive, HPC cluster, or other archival storage of the experimenter’s choice. These experiment-specific measurement plane channels are allocated from the FABRIC dataplane with QoS capability, or through the Internet, depending on the location of the archival site. In both cases, these channels are created in such a way that they are certain not to interfere with the experiment's data plane, effectively creating an experiment-specific measurement plane.

5.2.3 Provisioning for Packet Capture

MF provides the ability to capture packets of the experiment at line speed. To avoid interfering with the traffic on the data plane of the experiment, MF is able to instrument an experiment with a dedicated VM for the purposes of packet capture. In this case, the MF will use the CF to augment the slice by provisioning a VM with a high-speed NIC, connecting it to a mirror port on the data plane switch to be monitored, and configuring the data plane switch to mirror traffic from the target port to the mirror port . The VM needs to have sufficient memory/disk storage

space for processing and storing captured packets. These extra measurement resources must be accounted for and charged to the experiment using them.

Because the memory and disk resources of the packet capture VM are limited, packets can only be captured for a fixed amount of time before the VM runs out of space. If longer captures are desired, the captured packet will need to be moved out of the VM. In this case, the packet capture mechanisms need to be connected to additionally provisioned experimenter resources such as the ELK services described in section 5.2.1, and illustrated in Figure 6.

5.3 Timestamping Services

As we mentioned before, one of the salient features of Fabric MF is the ability to timestamp measurements with high-precision GPS-synchronized timestamps (a.k.a., *Packet GPS*) that provide accuracy across the FABRIC network to the tens of microseconds. The system clock of worker nodes in Fabric racks will be adjusted and synchronized with a PTP master clock, which is globally synchronized via the GPS signal. This clock signal can then be installed in the network interface cards (NIC) cards of the work nodes, which can then timestamp incoming packets with highly accurate timestamps. This requires running Linux PTP services. One is the *ptp4l* program that implements the PTP boundary clock and ordinary clock. Another is the *phc2sys* program that synchronizes the system clock to the PTP hardware clock on the network interface card (NIC). Setting the NIC clocks allows incoming packets to be time stamped by the NIC with extreme accuracy. While this approach may be useful to operators who are able to login to the worker nodes, it does not help experimenters running in VMs. Figure 5 illustrates the flow of a PTP signal within a FABRIC rack.

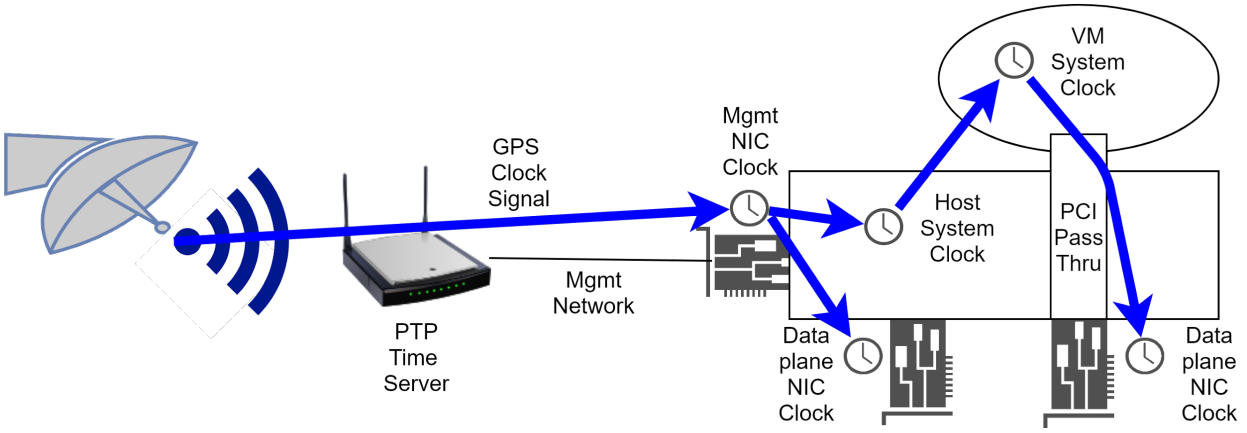


Figure 5: PTP Clock SIgnal to FABRIC worker/VM/NIC clocks

To provide experimenters with access to highly accurate timestamps, the MF must import the worker nodes' system clock values into the system clock of the VM. The MF utilizes a special Linux device module called *ptp_kvm* which is loaded upon the VM startup. This module

emulates a PTP Clock device whose time is kept in sync with the host system clock by the virtualization infrastructure. Once the system clock in the VM has been set, experimenters can use it to accurately timestamp their events. Likewise, any MF provisioned services (e.g., Beats/ELK and Prometheus exporters) can accurately timestamp the data they collect.

While the clock in the NIC cards can be set by the worker node, that does not help when NIC cards are mapped directly into a VM via PCI pass-through. In that case, *the ptp_kvm* clock in the VM must be used to set the clock in the NIC cards. Much like on the worker node, the MF services running in the VM will use the *phc2sys* to synchronize the VM *ptp_kvm* clock with the NIC cards that are mapped into the VM. Once synchronized, these cards can be used to accurately timestamp incoming packets. Note that the NIC cards are not capable of timestamping outgoing packets. Any outgoing traffic must be time stamped using the VM's system clock.

Note that *tcpdump* has an option (" -j *adapter_unsynced"*) that can be used to timestamp packets coming in from the NIC without any special programming of the NIC. However, tcpdump will timestamp outgoing packets with the system clock since the NIC cards FABRIC is using do not do outgoing packet timestamping.

5.4 Experiment-Specific MF components

For each VM in the experiment, the Measurement Points (MPs) -- packet capture, ELK beats, prometheus exporters can be enabled or disabled to collect user-specified VM-specific information. When packets arrive at a VM, the VM can be instrumented to collect certain parts (such as packet header) of the packets using tcpdump, possibly time-stamped with a PTP clock. We expect that the captured packets will first be stored locally at the VM and later be shipped to somewhere else for further processing and long-term archiving. Numeric measurement data collected by Prometheus exporters can also be temporarily stored on a local Prometheus database, which can be retrieved by Prometheus Querier and presented to the experimenters via the slice-specific Grafana GUI. Event/log messages/packet information will be published to the slice-specific ELK. One or multiple VMs need to be allocated for a slice to run the ELK system. One possible pipeline is shown in Figure 6. It collects data on two different NIC card MPs (possibly on different FABRIC racks) via tcpdump or Packetbeat and then forwards data to local logstash services to process/filter the data, which in turn forward the resulting data to an experiment-specific ELK system and web server that can be accessed by users that are part of the experiment, allowing them to visualize and work with their measurement data.

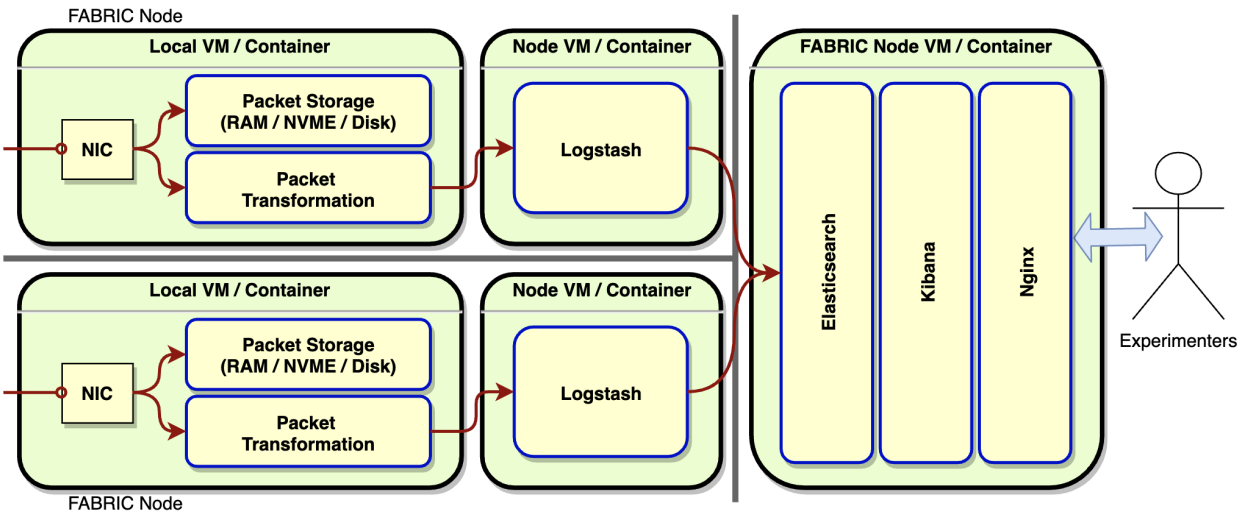


Figure 6: *MF Pipeline*: MP data capture, data filtering, processing and visualization services

5.5 Combining Experiment-specific measurements with Infrastructure Measurements

In addition to the experiment-specific measurement data, experimenters may also be interested in some information of the infrastructure their experiments are running on. We designed the MF Infrastructure Measurement Data Services to provide that function. The idea is to extract part of the infrastructure measurement data that can be shared with the specific experimenter without revealing any private information of other experiments. These data will be redirected to the experiment-specific measurement pipeline to make them available to the experimenters. By doing this, we can avoid double collecting the exact same information.

To achieve this, experimenters must authenticate to the MF Infrastructure Measurement service shown in Figure 4. With the information provided by the credential manager of the CF, the MF Infrastructure Measurement Data Service can determine what data the experimenter can have access to. Experimenters can also specify what specific infrastructure measurements they want to see. The MF Infrastructure Measurement Data Service reads the measurements from the global MF ELK or global MF Querier to continually get the data requested, and publishes them under slice-specific topics that can be consumed by subscriber processes running in the user's slice. (The details of how the pub/sub will be implemented are still to be worked out). For example, an experiment-specific Logstash can then subscribe to the topic to get the infrastructure data. The infrastructure data will then be combined with experiment-specific data to give users an integrated measurement data set that they can then run, analyze, process, and view.

5.6 Storing Experiment-specific Measurement Data

The size of experiment-specific measurement data can vary a lot, depending on the kinds of data collected. Event/log messages will be directed to the experiment-specific ES, while numeric measurements are stored in Prometheus/ThanOS storage. Line-speed packet capture on high-speed links provided by Fabric can only be stored in local storage because it operates at such a high speed and data grows so quickly that makes it infeasible to ship the raw data to remote locations. Because all these local resources may have limited capacity, experimenters will ultimately need to move the measurement data to external storage (e.g., commercial cloud, HPC DTNs, XSEDE resources, OSN, etc) for further processing. Fabric MF may add support in the future that would make it easy to make such transfers.

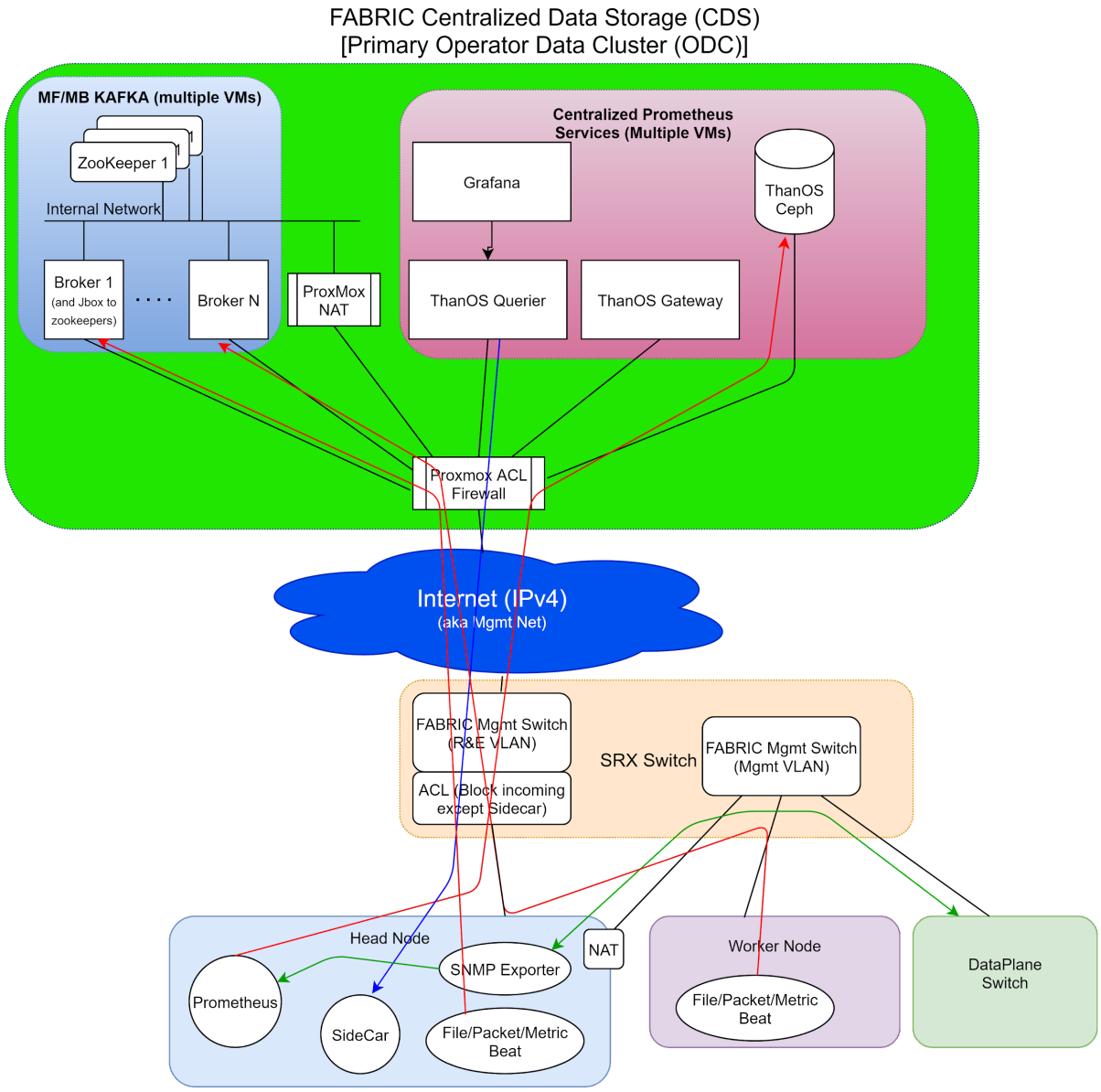
5.7 Visualizing Experiment-specific Measurement Data

Similar to the infrastructure measurement, experimenters can request the MF to provision and set up experiment-specific Kibana (as a part of ELK services) and Grafana to visualize the experiment-specific measurement data.

6. Security: Authentication, Authorization, Privacy

To use the MF Infrastructure Prometheus services, an operator has to login to Prometheus' Grafana UI using CI-Logon. The MF extracts information from Co-Manage to know the identity of the user and access rights based on the role of the user. To use the MF Infrastructure ELK services, an operator needs to be given an account on the MF-ELK Kibana server (Kibana's CI-Logon capabilities are not available via its free open-source version).

Communications between components will be secured by TLS and SASL using Fabric-issued certificates. ACLs are added to protect MF services from external Internet attacks, with most services visible to internal networks and worker nodes on Fabric racks.



7. Measurement Information Model

As noted in section 2, the MF and the CF are integrated in the sense that the MF can add extra information (essentially attributes) to any object in the CF information model. As a result, information about an MP will typically be recorded as an attribute (tag or JSON object) associated with a CF object in the information model. In some cases an MP might represent a complex service. For example, we might attach an MP to a network interface object in the information model that, in the model, appears to be part of the dataplane switch. However, when we implement that MP, we actually mirror the port to another port, create a VM with high speed NIC and lots of memory space to collect packets, and connect the new VM's NIC to the

mirror port. In other words, the implementation involves creating lots of extra measurement resources that are added to the user's experiment -- all to implement an MP on an interface in the original experiment.

Additional information about the FABRIC Information Model (FIM) can be found here.

8. MF API

Users interact with the MF via both the graphical user interface and API calls. We will design API calls (described elsewhere) to turn on/off measurements via the instrumentation and control services, write out event/log messages with highly accurate timestamps, access measurement data and download graphs of the data. Details of the MF API are still being finalized.

9. MF and the Message Bus

MF communicates with the Fabric CF via the message bus. The MF Instrumentation and Control Service allocates MP resources for an experiment and turns on/off the MPs in user slices. The MF may need to authenticate users and get tokens about access rights of users via the Message Bus. Components of MF may also use the message bus for control and coordination.

Support

FABRIC is supported in part by a Mid-Scale RI-1 NSF award under Grant No. 1935966.