US 20190230127A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2019/0230127 A1

Gandham et al. (43) Pub. Date: Jul. 25, 2019

(54) SECURE PUBLISHING FOR POLICY UPDATES

(71) Applicant: **Cisco Technology, Inc.**, San Jose, CA (US)

(72) Inventors: **Shashi Gandham**, Fremont, CA (US); **Navindra Yadav**, Cupertino, CA (US); **Janardhanan Radhakrishnan**, Dublin, CA (US); **Hoang-Nam Nguyen**, San Jose, CA (US); **Umesh Paul Mahindra**, Cupertino, CA (US); **Sunil Gupta**, Milpitas, CA (US); **Praneeth Vallem**, San Jose, CA (US); **Supreeth Rao**, Cupertino, CA (US); **Darshan Shrinath Purandare**, Fremont, CA (US); **Xuan Zou**, Sunnyvale, CA (US); **Girish Anant Kalele**, Monte Sereno, CA (US); **Jothi Prakash Prabakaran**, Fremont, CA (US)

(21) Appl. No.: **16/032,765**

(22) Filed: **Jul. 11, 2018**

**Related U.S. Application Data**

(60) Provisional application No. 62/621,900, filed on Jan. 25, 2018.

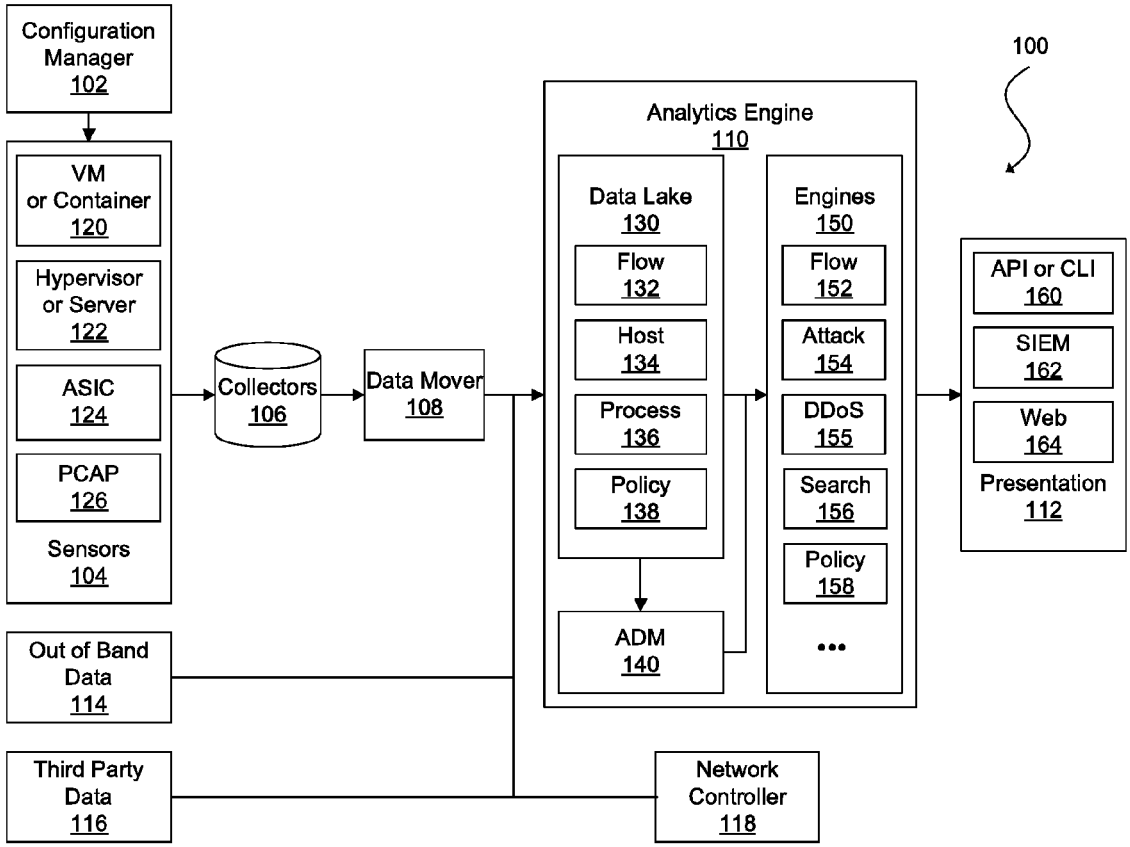
**Publication Classification**

(51) **Int. Cl.**
***H04L 29/06*** (2006.01)
***H04L 12/24*** (2006.01)

(52) **U.S. Cl.**
CPC ..... ***H04L 63/20*** (2013.01); *H04L 67/10* (2013.01); ***H04L 41/0806*** (2013.01); ***H04L 63/0823*** (2013.01)

(57) **ABSTRACT**

Aspects of the disclosed technology relate to ways to authenticate customer/subscriber access to a policy update stream. A process of the technology can include steps for instantiating a network monitoring device in response to a request, the request comprising one or more configuration parameters for the network monitoring device, and receiving a first certificate from the network monitoring device, wherein the first certificate is based on the one or more configuration parameters. In some aspects, the steps can further include sending the first certificate to a processing pipeline for authentication, wherein the processing pipeline is configured to authenticate the first certificate based on a second certificate received by the processing pipeline from the network monitoring device. Systems and machine readable media are also provided.



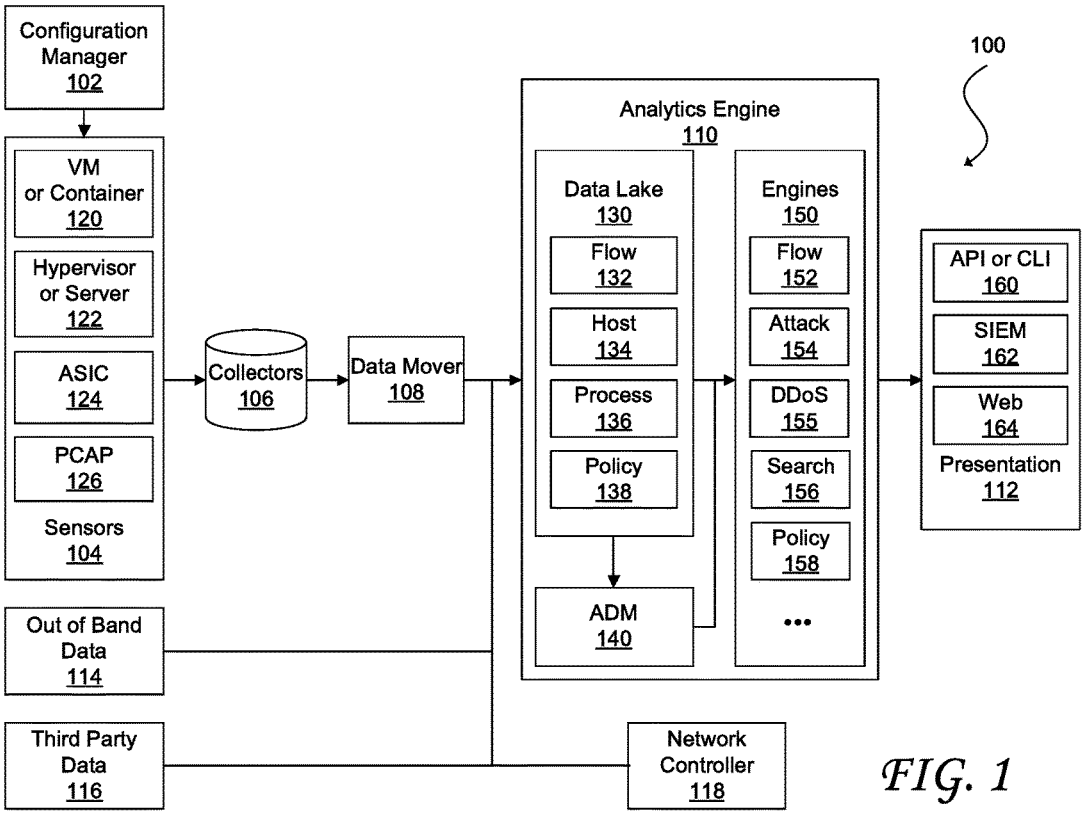

Configuration Manager 102
VM or Container 120
Hypervisor or Server 122
ASIC 124
PCAP 126
Sensors 104
Collectors 106
Data Mover 108
Analytics Engine 110
Data Lake 130
Flow 132
Host 134
Process 136
Policy 138
ADM 140
Engines 150
Flow 152
Attack 154
DDoS 155
Search 156
Policy 158
...
100
API or CLI 160
SIEM 162
Web 164
Presentation 112
Out of Band Data 114
Third Party Data 116
Network Controller 118

FIG. 1

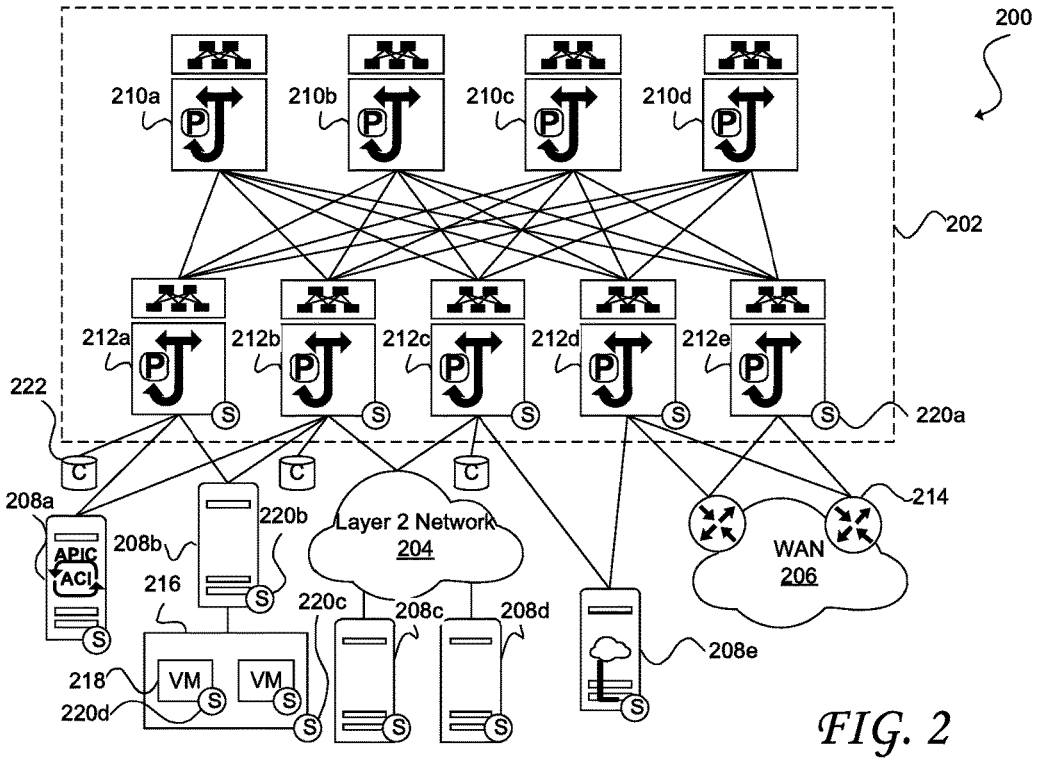

200
210a
210b
210c
210d
202
212a
212b
212c
212d
212e
222
220a
208a
208b
220b
216
Layer 2 Network
204
214
WAN
206
220c
208c
208d
208e
218
220d
VM
VM
APIC
ACI

FIG. 2

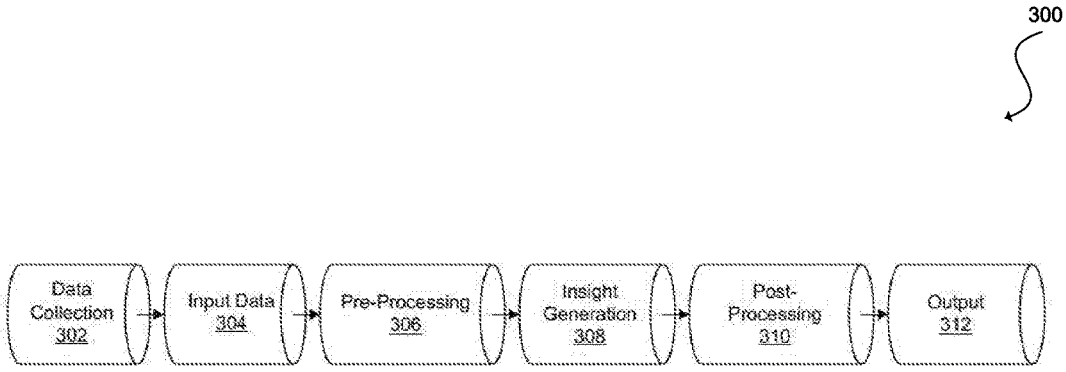

300
Data Collection 302
Input Data 304
Pre-Processing 306
Insight Generation 308
Post-Processing 310
Output 312

*FIG. 3*

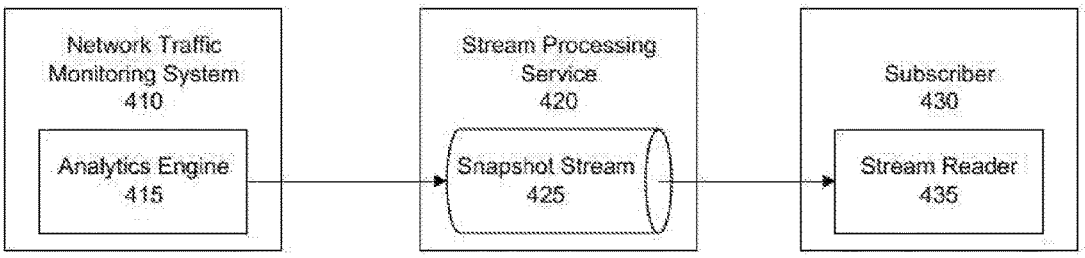

Network Traffic Monitoring System 410
Analytics Engine 415
Stream Processing Service 420
Snapshot Stream 425
Subscriber 430
Stream Reader 435

*FIG. 4*

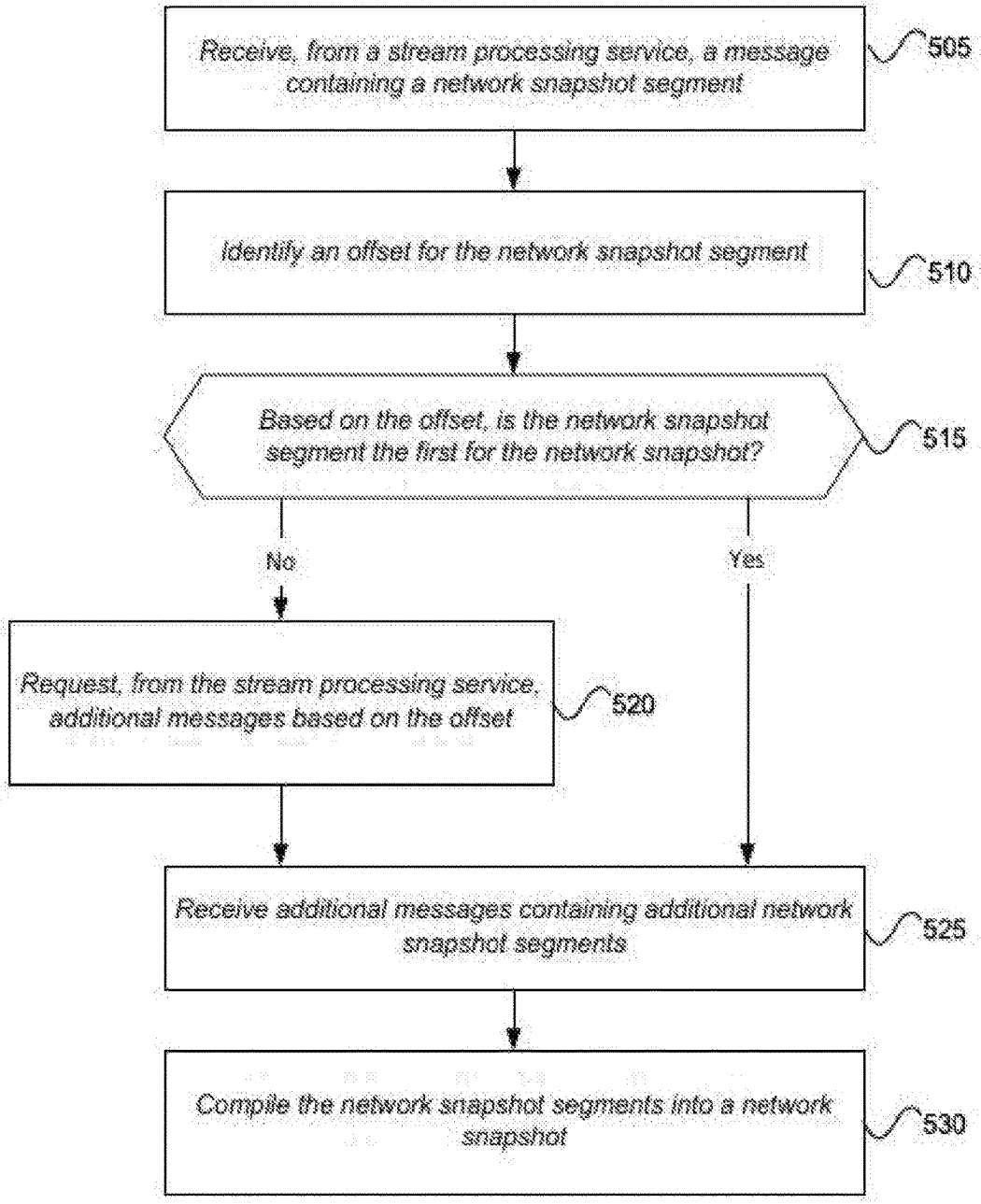

Receive, from a stream processing service, a message containing a network snapshot segment
505
Identify an offset for the network snapshot segment
510
Based on the offset, is the network snapshot segment the first for the network snapshot?
515
No
Yes
Request, from the stream processing service, additional messages based on the offset
520
Receive additional messages containing additional network snapshot segments
525
Compile the network snapshot segments into a network snapshot
530

FIG. 5

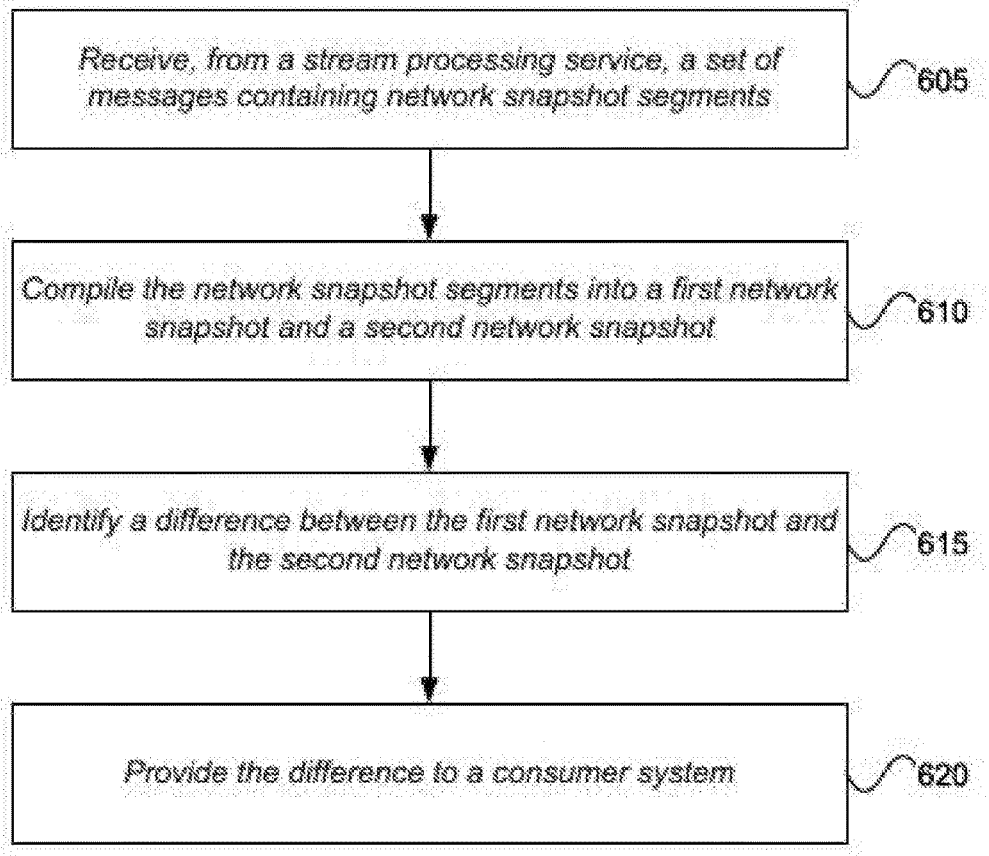

Receive, from a stream processing service, a set of messages containing network snapshot segments
605
Compile the network snapshot segments into a first network snapshot and a second network snapshot
610
Identify a difference between the first network snapshot and the second network snapshot
615
Provide the difference to a consumer system
620

*FIG. 6*

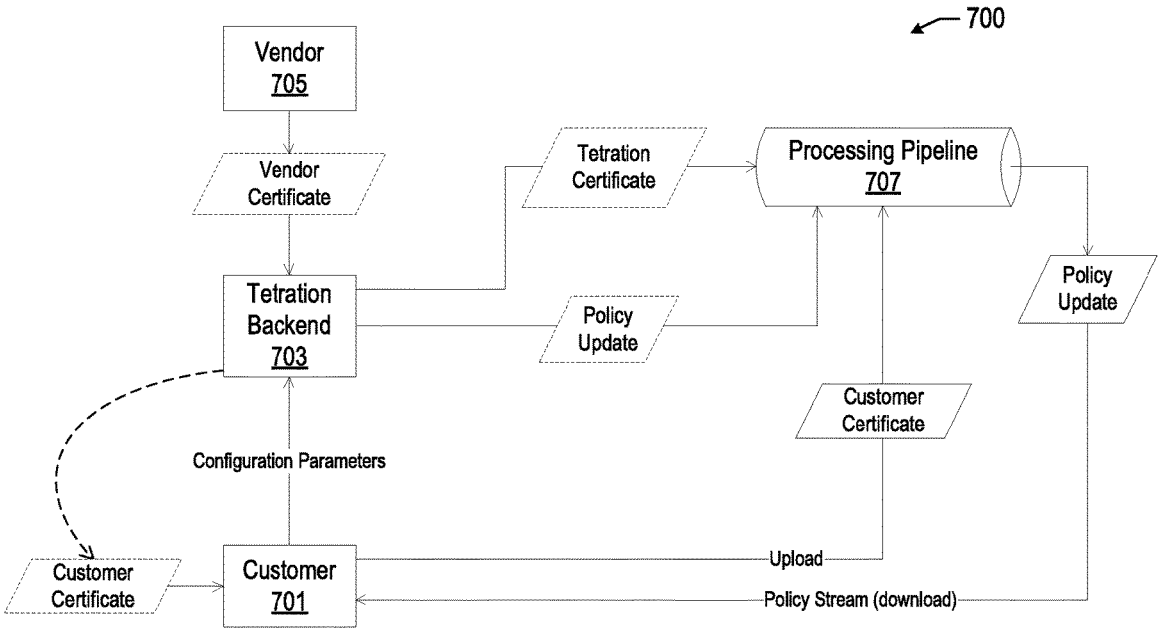

700
Vendor
705
Vendor
Certificate
Tetration
Backend
703
Tetration
Certificate
Processing Pipeline
707
Policy
Update
Policy
Update
Customer
Certificate
Configuration Parameters
Customer
Certificate
Customer
701
Upload
Policy Stream (download)

*FIG. 7*

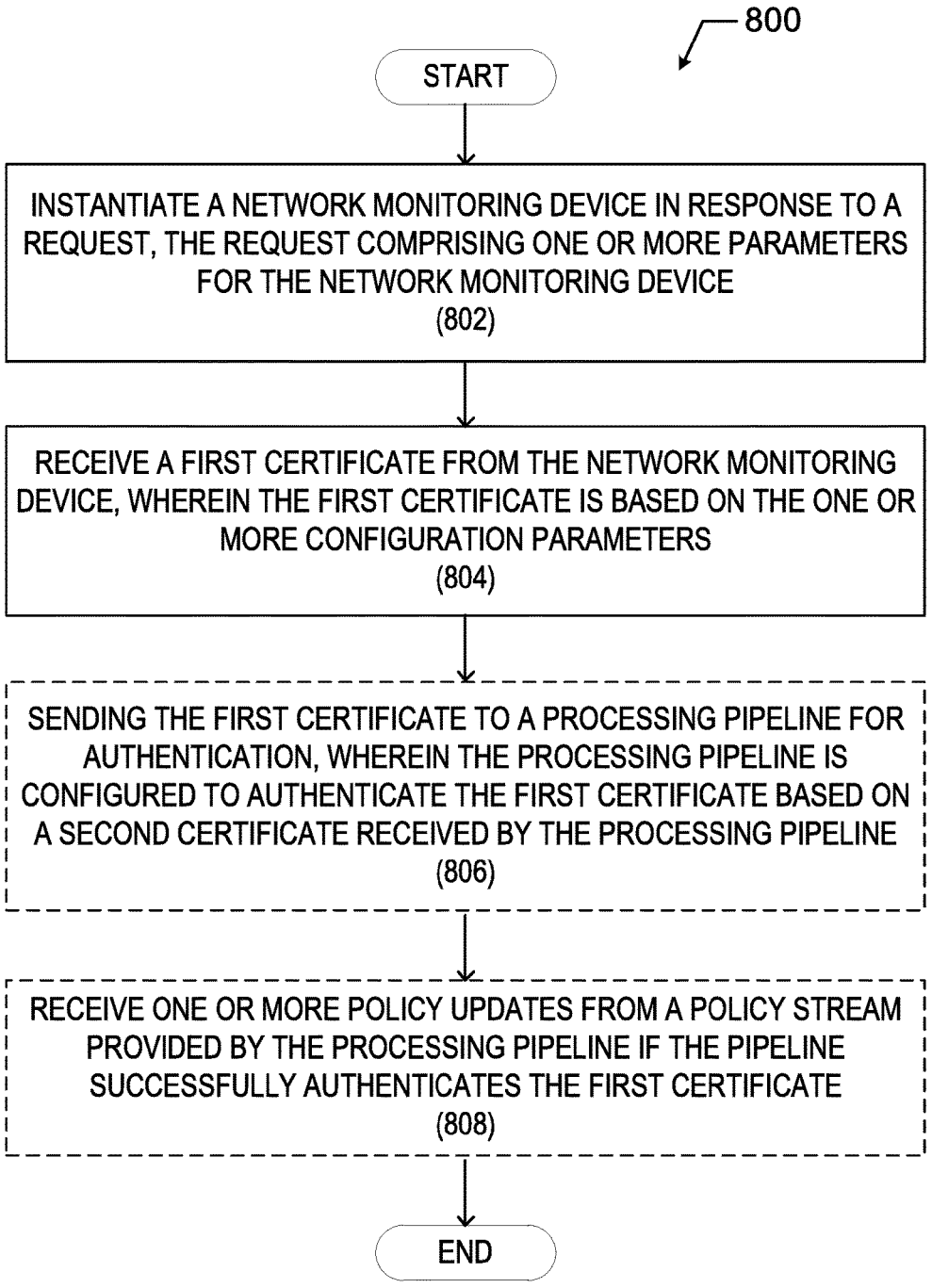

800
START
INSTANTIATE A NETWORK MONITORING DEVICE IN RESPONSE TO A REQUEST, THE REQUEST COMPRISING ONE OR MORE PARAMETERS FOR THE NETWORK MONITORING DEVICE
(802)
RECEIVE A FIRST CERTIFICATE FROM THE NETWORK MONITORING DEVICE, WHEREIN THE FIRST CERTIFICATE IS BASED ON THE ONE OR MORE CONFIGURATION PARAMETERS
(804)
SENDING THE FIRST CERTIFICATE TO A PROCESSING PIPELINE FOR AUTHENTICATION, WHEREIN THE PROCESSING PIPELINE IS CONFIGURED TO AUTHENTICATE THE FIRST CERTIFICATE BASED ON A SECOND CERTIFICATE RECEIVED BY THE PROCESSING PIPELINE
(806)
RECEIVE ONE OR MORE POLICY UPDATES FROM A POLICY STREAM PROVIDED BY THE PROCESSING PIPELINE IF THE PIPELINE SUCCESSFULLY AUTHENTICATES THE FIRST CERTIFICATE
(808)
END

*FIG. 8*

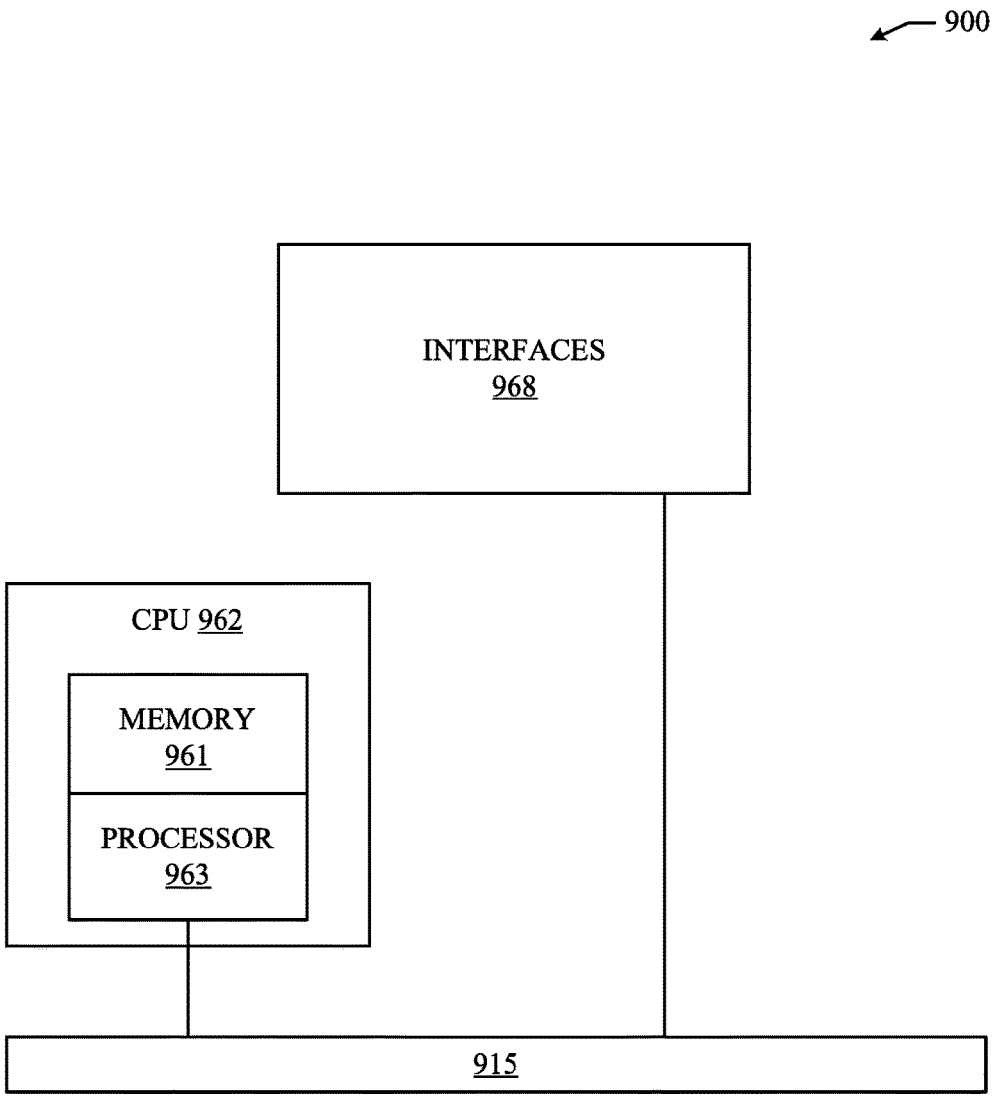



*FIG. 9*

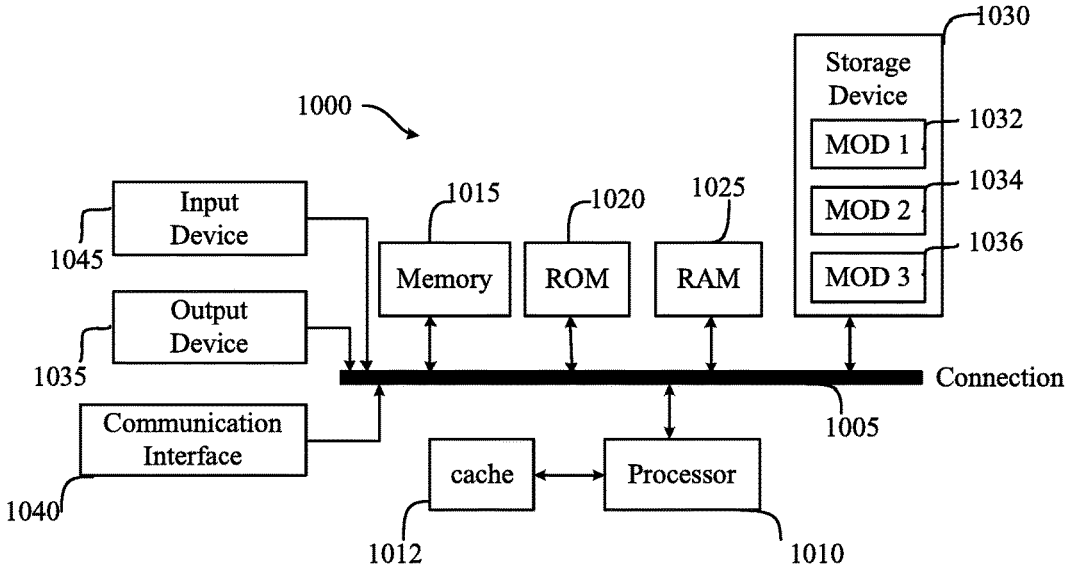

1000
1030
Storage Device
1032
MOD 1
1034
MOD 2
1036
MOD 3
1015
1020
1025
Memory
ROM
RAM
Input Device
1045
Output Device
1035
Communication Interface
1040
Connection
1005
cache
1012
Processor
1010

*FIG. 10*

# SECURE PUBLISHING FOR POLICY UPDATES

## CROSS-REFERENCE TO RELATED APPLICATION

**[0001]** This application claims priority to U.S. Provisional Application 62/621,900, filed on Jan. 25, 2018, entitled "SECURE PUBLISHING OF NETWORK POLICIES," the content of which is incorporated herein by reference in its entirety.

## TECHNICAL FIELD

**[0002]** The subject matter of this disclosure relates in general to the field of computer networks, and more specifically to efficiently transferring large amounts of data generated by systems managing a network.

## BACKGROUND

**[0003]** An enterprise application is a set of workloads (e.g., computing, networking, and storage) that are generally distributed across various nodes (or endpoints) of a network and the relationships (e.g., connectivity, dependencies, network and security policies, etc.) between the workloads. A typical application may include a presentation tier, an application tier, and a data tier. The presentation tier may depend on the application tier and authentication services, and the application tier may depend on the web tier and external network services (e.g., a travel reservation system, an ordering tool, a billing tool, etc.). These tiers may further depend on firewall, load balancing, wide area network (WAN) acceleration, and other network services. An enterprise can include hundreds or thousands of applications of similar and different architectures.

**[0004]** An expansive or thorough understanding of a data center and applications running in the data center can be critical for network management tasks such as anomaly detection (e.g., network attacks and misconfiguration), asset management (e.g., monitoring, capacity planning, consolidation, migration, and continuity planning), and compliance (e.g. conformance with governmental regulations, industry standards, and corporate policies). Despite the complexities of the interrelationships among workloads discussed above, the many approaches for developing insight into an enterprise's workloads require comprehensive knowledge on the part of human operators and processes that are manual and largely customized for a particular enterprise.

## BRIEF DESCRIPTION OF THE FIGURES

**[0005]** In order to describe the manner in which the above-recited and other advantages and features of the disclosure can be obtained, a more particular description of the principles briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. Understanding that these drawings depict only embodiments of the disclosure and are not therefore to be considered to be limiting of its scope, the principles herein are described and explained with additional specificity and detail through the use of the accompanying drawings in which:

**[0006]** FIG. **1** illustrates an example of a network traffic monitoring system, in accordance with an embodiment;

**[0007]** FIG. **2** illustrates an example of a network environment, in accordance with an embodiment;

**[0008]** FIG. **3** illustrates an example of a data pipeline for generating network insights based on collected network information, in accordance with an embodiment;

**[0009]** FIG. **4** illustrates an example of a network traffic monitoring system providing network snapshots to subscribers, in accordance with an embodiment;

**[0010]** FIG. **5** illustrates an example of a process for compiling a network snapshot, in accordance with an embodiment;

**[0011]** FIG. **6** illustrates an example of a process for identifying a difference between network snapshots, in accordance with an embodiment;

**[0012]** FIG. **7** illustrates an example architecture for implementing a secure policy update publishing stream, according to some aspects of the technology;

**[0013]** FIG. **8** illustrates an example process for authenticating customer access to a secure policy stream, for receiving network policy updates, according to some aspects of the technology;

**[0014]** FIG. **9** illustrates an example of a processor based networking device that can be used for implementing some aspects of the disclosed technology; and

**[0015]** FIG. **10** illustrates an example processing device that may be used to implement some aspects of the technology.

## DETAILED DESCRIPTION

**[0016]** The detailed description set forth below is intended as a description of various configurations of embodiments and is not intended to represent the only configurations in which the subject matter of this disclosure can be practiced. The appended drawings are incorporated herein and constitute a part of the detailed description. The detailed description includes specific details for the purpose of providing a more thorough understanding of the subject matter of this disclosure. However, it will be clear and apparent that the subject matter of this disclosure is not limited to the specific details set forth herein and may be practiced without these details. In some instances, structures and components are shown in block diagram form in order to avoid obscuring the concepts of the subject matter of this disclosure.

**[0017]** Overview:

**[0018]** Aspects of the disclosed technology relate to ways to authenticate customer/subscriber access to a policy update stream. A process of the technology can include steps for instantiating a network monitoring device in response to a request, the request comprising one or more configuration parameters for the network monitoring device, and receiving a first certificate from the network monitoring device, wherein the first certificate is based on the one or more configuration parameters. In some aspects, the steps can further include sending the first certificate to a processing pipeline for authentication, wherein the processing pipeline is configured to authenticate the first certificate based on a second certificate received by the processing pipeline from the network monitoring device. Systems and machine readable media are also provided.

**[0019]** In some implementations, a process implementing the disclosed technology could include additional steps for receiving one or more policy updates from a policy stream provided by the processing pipeline if the processing pipeline successfully authenticates the first certificate, decrypting the one or more policy updates received from the policy

stream, and implementing at least one change indicated by the one or more policy updates to an associated customer network.

## DESCRIPTION

**[0020]** Various embodiments of the disclosure are discussed in detail below. While specific implementations are discussed, it should be understood that this is done for illustrative purposes only. A person skilled in the relevant art will recognize that other components and configurations may be used without departing from the spirit and scope of the disclosure.

**[0021]** Sensors deployed in a network can be used to gather network information related to network traffic of nodes operating in the network and process information for nodes and applications running in the network. Gathered network information can be analyzed to provide insights into the operation of the nodes in the network, otherwise referred to as analytics. In particular, discovered application or inventories, application dependencies, policies, efficiencies, resource and bandwidth usage, and network flows can be determined for the network using the network traffic data. For example, an analytics engine can be configured to automate discovery of applications running in the network, map the applications' interdependencies, or generate a set of proposed network policies for implementation. For example, new network policies may be created and published, e.g., via a secure policy stream, for properly authenticated users, e.g., customers or "subscribers."

**[0022]** However, one problem in providing network policy updates is to ensure that network intents can only be read and processed by authorized customers and third-party vendor applications. Securing network policy updates is especially important for preventing attackers from modifying/overwriting intents in order to create backdoors, for example, by opening ports that can be used to launch cyber-attacks.

**[0023]** One limitation of conventional network policy update processes is that the updates are un-encrypted and/or that the credentials used to authenticate the receiving party only rely on a single authentication certificate. These weak security protocols jeopardize the policy update process of conventional networks by leaving them open to potential man-in-the-middle attacks. For example, a nefarious user may only need to find the correct channel on which unencrypted policy updates are provided to compromise the network. Alternatively, by compromising a single system (e.g., the network management cluster, or vendor certificate), policy updates may be compromised in systems using single-certificate authentication procedures.

**[0024]** Aspects of the disclosed technology address the foregoing problems of policy security by ensuring authorization of policy subscriber (customer) and third-party vendor applications using a Tetration inventory manager to identify network appliances based on their addresses or corresponding proxy server. In some aspects, authorization requires permitted systems to be known by the Tetration inventory manager, for example, using an automatic (or manual) registration mechanism.

**[0025]** In some implementations, the publisher (e.g., Tetration cluster) can create a secure message queuing channel that is configured such that only permitted applications/customers with validated client certificates to access the channel to receive policy updates. In some implementations, the publisher can issue a customer/subscriber certificate upon creation/instantiation of the monitoring cluster (publisher). Additionally the publisher, can independently provide a corresponding certificate to the secure message queuing channel (e.g., Kafka), that can be used to validate the customer certificate. As an added layer of security, policy updates can encrypted before being pushed through the queuing channel

**[0026]** Referring now to the drawings, FIG. **1** is an illustration of a network traffic monitoring system **100** in accordance with an embodiment. The network traffic monitoring system **100** can include a configuration manager **102**, sensors **104**, a collector module **106**, a data mover module **108**, an analytics engine **110**, and a presentation module **112**. In FIG. **1**, the analytics engine **110** is also shown in communication with out-of-band data sources **114**, third party data sources **116**, and a network controller **118**.

**[0027]** The configuration manager **102** can be used to provision and maintain the sensors **104**, including installing sensor software or firmware in various nodes of a network, configuring the sensors **104**, updating the sensor software or firmware, among other sensor management tasks. For example, the sensors **104** can be implemented as virtual partition images (e.g., virtual machine (VM) images or container images), and the configuration manager **102** can distribute the images to host machines. In general, a virtual partition may be an instance of a VM, container, sandbox, or other isolated software environment. The software environment may include an operating system and application software. For software running within a virtual partition, the virtual partition may appear to be, for example, one of many servers or one of many operating systems executed on a single physical server. The configuration manager **102** can instantiate a new virtual partition or migrate an existing partition to a different physical server. The configuration manager **102** can also be used to configure the new or migrated sensor.

**[0028]** The configuration manager **102** can monitor the health of the sensors **104**. For example, the configuration manager **102** may request for status updates and/or receive heartbeat messages, initiate performance tests, generate health checks, and perform other health monitoring tasks. In some embodiments, the configuration manager **102** can also authenticate the sensors **104**. For instance, the sensors **104** can be assigned a unique identifier, such as by using a one-way hash function of a sensor's basic input/out system (BIOS) universally unique identifier (UUID) and a secret key stored by the configuration image manager **102**. The UUID can be a large number that may be difficult for a malicious sensor or other device or component to guess. In some embodiments, the configuration manager **102** can keep the sensors **104** up to date by installing the latest versions of sensor software and/or applying patches. The configuration manager **102** can obtain these updates automatically from a local source or the Internet.

**[0029]** The sensors **104** can reside on various nodes of a network, such as a virtual partition (e.g., VM or container) **120**; a hypervisor or shared kernel managing one or more virtual partitions and/or physical servers **122**, an application-specific integrated circuit (ASIC) **124** of a switch, router, gateway, or other networking device, or a packet capture (pcap) **126** appliance (e.g., a standalone packet monitor, a device connected to a network devices monitoring port, a device connected in series along a main trunk of a datacen-

ter, or similar device), or other element of a network. The sensors **104** can monitor network traffic between nodes, and send network traffic data and corresponding data (e.g., host data, process data, user data, etc.) to the collectors **106** for storage. For example, the sensors **104** can sniff packets being sent over its hosts' physical or virtual network interface card (NIC), or individual processes can be configured to report network traffic and corresponding data to the sensors **104**. Incorporating the sensors **104** on multiple nodes and within multiple partitions of some nodes of the network can provide for robust capture of network traffic and corresponding data from each hop of data transmission. In some embodiments, each node of the network (e.g., VM, container, or other virtual partition (e.g., container) **120**, hypervisor, shared kernel, or physical server **122**, ASIC **124**, pcap **126**, etc.) includes a respective sensor **104**. However, it should be understood that various software and hardware configurations can be used to implement the sensor network **104**.

**[0030]** As the sensors **104** capture communications and corresponding data, they may continuously send network traffic data to the collectors **106**. The network traffic data can include metadata relating to a packet, a collection of packets, a flow, a bidirectional flow, a group of flows, a session, or a network communication of another granularity. That is, the network traffic data can generally include any information describing communication on all layers of the Open Systems Interconnection (OSI) model. For example, the network traffic data can include source/destination MAC address, source/destination IP address, protocol, port number, etc. In some embodiments, the network traffic data can also include summaries of network activity or other network statistics such as number of packets, number of bytes, number of flows, bandwidth usage, response time, latency, packet loss, jitter, and other network statistics.

**[0031]** The sensors **104** can also determine additional data for each session, bidirectional flow, flow, packet, or other more granular or less granular network communication. The additional data can include host and/or endpoint information, virtual partition information, sensor information, process information, user information, tenant information, application information, network topology, application dependency mapping, cluster information, or other information corresponding to each flow.

**[0032]** In some embodiments, the sensors **104** can perform some preprocessing of the network traffic and corresponding data before sending the data to the collectors **106**. For example, the sensors **104** can remove extraneous or duplicative data or they can create summaries of the data (e.g., latency, number of packets per flow, number of bytes per flow, number of flows, etc.). In some embodiments, the sensors **104** can be configured to only capture certain types of network information and disregard the rest. In some embodiments, the sensors **104** can be configured to capture only a representative sample of packets (e.g., every 1,000th packet or other suitable sample rate) and corresponding data.

**[0033]** Since the sensors **104** may be located throughout the network, network traffic and corresponding data can be collected from multiple vantage points or multiple perspectives in the network to provide a more comprehensive view of network behavior. The capture of network traffic and corresponding data from multiple perspectives rather than just at a single sensor located in the data path or in communication with a component in the data path, allows the data to be correlated from the various data sources, which may be used as additional data points by the analytics engine **110**. Further, collecting network traffic and corresponding data from multiple points of view ensures more accurate data is captured. For example, other types of sensor networks may be limited to sensors running on external-facing network devices (e.g., routers, switches, network appliances, etc.) such that east-west traffic, including VM-to-VM or container-to-container traffic on a same host, may not be monitored. In addition, packets that are dropped before traversing a network device or packets containing errors may not be accurately monitored by other types of sensor networks. The sensor network **104** of various embodiments substantially mitigates or eliminates these issues altogether by locating sensors at multiple points of potential failure. Moreover, the network traffic monitoring system **100** can verify multiple instances of data for a flow (e.g., source endpoint flow data, network device flow data, and endpoint flow data) against one another.

**[0034]** In some embodiments, the network traffic monitoring system **100** can assess a degree of accuracy of flow data sets from multiple sensors and utilize a flow data set from a single sensor determined to be the most accurate and/or complete. The degree of accuracy can be based on factors such as network topology (e.g., a sensor closer to the source may be more likely to be more accurate than a sensor closer to the destination), a state of a sensor or a node hosting the sensor (e.g., a compromised sensor/node may have less accurate flow data than an uncompromised sensor/node), or flow data volume (e.g., a sensor capturing a greater number of packets for a flow may be more accurate than a sensor capturing a smaller number of packets).

**[0035]** In some embodiments, the network traffic monitoring system **100** can assemble the most accurate flow data set and corresponding data from multiple sensors. For instance, a first sensor along a data path may capture data for a first packet of a flow but may be missing data for a second packet of the flow while the situation is reversed for a second sensor along the data path. The network traffic monitoring system **100** can assemble data for the flow from the first packet captured by the first sensor and the second packet captured by the second sensor.

**[0036]** As discussed, the sensors **104** can send network traffic and corresponding data to the collectors **106**. In some embodiments, each sensor can be assigned to a primary collector and a secondary collector as part of a high availability scheme. If the primary collector fails or communications between the sensor and the primary collector are not otherwise possible, a sensor can send its network traffic and corresponding data to the secondary collector. In other embodiments, the sensors **104** are not assigned specific collectors but the network traffic monitoring system **100** can determine an optimal collector for receiving the network traffic and corresponding data through a discovery process. In such embodiments, a sensor can change where it sends it network traffic and corresponding data if its environments changes, such as if a default collector fails or if the sensor is migrated to a new location and it would be optimal for the sensor to send its data to a different collector. For example, it may be preferable for the sensor to send its network traffic and corresponding data on a particular path and/or to a particular collector based on latency, shortest path, monetary cost (e.g., using private resources versus a public resources provided by a public cloud provider), error rate, or some combination of these factors. In other embodiments, a sensor

can send different types of network traffic and corresponding data to different collectors. For example, the sensor can send first network traffic and corresponding data related to one type of process to one collector and second network traffic and corresponding data related to another type of process to another collector.

**[0037]** The collectors **106** can be any type of storage medium that can serve as a repository for the network traffic and corresponding data captured by the sensors **104**. In some embodiments, data storage for the collectors **106** is located in an in-memory database, such as dashDB from IBM®, although it should be appreciated that the data storage for the collectors **106** can be any software and/or hardware capable of providing rapid random access speeds typically used for analytics software. In various embodiments, the collectors **106** can utilize solid state drives, disk drives, magnetic tape drives, or a combination of the foregoing according to cost, responsiveness, and size requirements. Further, the collectors **106** can utilize various database structures such as a normalized relational database or a NoSQL database, among others.

**[0038]** In some embodiments, the collectors **106** may only serve as network storage for the network traffic monitoring system **100**. In such embodiments, the network traffic monitoring system **100** can include a data mover module **108** for retrieving data from the collectors **106** and making the data available to network clients, such as the components of the analytics engine **110**. In effect, the data mover module **108** can serve as a gateway for presenting network-attached storage to the network clients. In other embodiments, the collectors **106** can perform additional functions, such as organizing, summarizing, and preprocessing data. For example, the collectors **106** can tabulate how often packets of certain sizes or types are transmitted from different nodes of the network. The collectors **106** can also characterize the traffic flows going to and from various nodes. In some embodiments, the collectors **106** can match packets based on sequence numbers, thus identifying traffic flows and connection links. As it may be inefficient to retain all data indefinitely in certain circumstances, in some embodiments, the collectors **106** can periodically replace detailed network traffic data with consolidated summaries. In this manner, the collectors **106** can retain a complete dataset describing one period (e.g., the past minute or other suitable period of time), with a smaller dataset of another period (e.g., the previous 2-10 minutes or other suitable period of time), and progressively consolidate network traffic and corresponding data of other periods of time (e.g., day, week, month, year, etc.). In some embodiments, network traffic and corresponding data for a set of flows identified as normal or routine can be winnowed at an earlier period of time while a more complete data set may be retained for a lengthier period of time for another set of flows identified as anomalous or as an attack.

**[0039]** Computer networks may be exposed to a variety of different attacks that expose vulnerabilities of computer systems in order to compromise their security. Some network traffic may be associated with malicious programs or devices. The analytics engine **110** may be provided with examples of network states corresponding to an attack and network states corresponding to normal operation. The analytics engine **110** can then analyze network traffic and corresponding data to recognize when the network is under attack. In some embodiments, the network may operate within a trusted environment for a period of time so that the analytics engine **110** can establish a baseline of normal operation. Since malware is constantly evolving and changing, machine learning may be used to dynamically update models for identifying malicious traffic patterns.

**[0040]** In some embodiments, the analytics engine **110** may be used to identify observations which differ from other examples in a dataset. For example, if a training set of example data with known outlier labels exists, supervised anomaly detection techniques may be used. Supervised anomaly detection techniques utilize data sets that have been labeled as normal and abnormal and train a classifier. In a case in which it is unknown whether examples in the training data are outliers, unsupervised anomaly techniques may be used. Unsupervised anomaly detection techniques may be used to detect anomalies in an unlabeled test data set under the assumption that the majority of instances in the data set are normal by looking for instances that seem to fit to the remainder of the data set.

**[0041]** The analytics engine **110** can include a data lake **130**, an application dependency mapping (ADM) module **140**, and elastic processing engines **150**. The data lake **130** is a large-scale storage repository that provides massive storage for various types of data, enormous processing power, and the ability to handle nearly limitless concurrent tasks or jobs. In some embodiments, the data lake **130** is implemented using the Hadoop® Distributed File System (HDFS™) from Apache® Software Foundation of Forest Hill, Md. HDFS™ is a highly scalable and distributed file system that can scale to thousands of cluster nodes, millions of files, and petabytes of data. HDFS™ is optimized for batch processing where data locations are exposed to allow computations to take place where the data resides. HDFS™ provides a single namespace for an entire cluster to allow for data coherency in a write-once, read-many access model. That is, clients can only append to existing files in the node. In HDFS™, files are separated into blocks, which are typically 64 MB in size and are replicated in multiple data nodes. Clients access data directly from data nodes.

**[0042]** In some embodiments, the data mover **108** receives raw network traffic and corresponding data from the collectors **106** and distributes or pushes the data to the data lake **130**. The data lake **130** can also receive and store out-of-band data **114**, such as statuses on power levels, network availability, server performance, temperature conditions, cage door positions, and other data from internal sources, and third party data **116**, such as security reports (e.g., provided by Cisco® Systems, Inc. of San Jose, Calif., Arbor Networks® of Burlington, Mass., Symantec® Corp. of Sunnyvale, Calif., Sophos® Group plc of Abingdon, England, Microsoft® Corp. of Seattle, Wash., Verizon® Communications, Inc. of New York, N.Y., among others), geolocation data, IP watch lists, Whois data, configuration management database (CMDB) or configuration management system (CMS) as a service, and other data from external sources. In other embodiments, the data lake **130** may instead fetch or pull raw traffic and corresponding data from the collectors **106** and relevant data from the out-of-band data sources **114** and the third party data sources **116**. In yet other embodiments, the functionality of the collectors **106**, the data mover **108**, the out-of-band data sources **114**, the third party data sources **116**, and the data lake **130** can be combined. Various combinations and configurations are possible as would be known to one of ordinary skill in the art.

**[0043]** Each component of the data lake **130** can perform certain processing of the raw network traffic data and/or other data (e.g., host data, process data, user data, out-of-band data or third party data) to transform the raw data to a form useable by the elastic processing engines **150**. In some embodiments, the data lake **130** can include repositories for flow attributes **132**, host and/or endpoint attributes **134**, process attributes **136**, and policy attributes **138**. In some embodiments, the data lake **130** can also include repositories for VM or container attributes, application attributes, tenant attributes, network topology, application dependency maps, cluster attributes, etc.

**[0044]** Flow attributes **132** relate to information about flows traversing the network. A flow is generally one or more packets sharing certain attributes that are sent within a network within a specified period of time. The flow attributes **132** can include packet header fields such as a source address (e.g., Internet Protocol (IP) address, Media Access Control (MAC) address, Domain Name System (DNS) name, or other network address), source port, destination address, destination port, protocol type, class of service, among other fields. The source address may correspond to a first endpoint (e.g., network device, physical server, virtual partition, etc.) of the network, and the destination address may correspond to a second endpoint, a multicast group, or a broadcast domain. The flow attributes **132** can also include aggregate packet data such as flow start time, flow end time, number of packets for a flow, number of bytes for a flow, the union of TCP flags for a flow, among other flow data.

**[0045]** The host and/or endpoint attributes **134** describe host and/or endpoint data for each flow, and can include host and/or endpoint name, network address, operating system, CPU usage, network usage, disk space, ports, logged users, scheduled jobs, open files, and information regarding files and/or directories stored on a host and/or endpoint (e.g., presence, absence, or modifications of log files, configuration files, device special files, or protected electronic information). As discussed, in some embodiments, the host and/or endpoints attributes **134** can also include the out-of-band data **114** regarding hosts such as power level, temperature, and physical location (e.g., room, row, rack, cage door position, etc.) or the third party data **116** such as whether a host and/or endpoint is on an IP watch list or otherwise associated with a security threat, Whois data, or geocoordinates. In some embodiments, the out-of-band data **114** and the third party data **116** may be associated by process, user, flow, or other more granular or less granular network element or network communication.

**[0046]** The process attributes **136** relate to process data corresponding to each flow, and can include process name (e.g., bash, httpd, netstat, etc.), ID, parent process ID, path (e.g., /usr2/usemame/bin/, /usr/local/bin, /usr/bin, etc.), CPU utilization, memory utilization, memory address, scheduling information, nice value, flags, priority, status, start time, terminal type, CPU time taken by the process, the command that started the process, and information regarding a process owner (e.g., user name, ID, user's real name, e-mail address, user's groups, terminal information, login time, expiration date of login, idle time, and information regarding files and/or directories of the user).

**[0047]** The policy attributes **138** contain information relating to network policies. Policies establish whether a particular flow is allowed or denied by the network as well as a specific route by which a packet traverses the network. Policies can also be used to mark packets so that certain kinds of traffic receive differentiated service when used in combination with queuing techniques such as those based on priority, fairness, weighted fairness, token bucket, random early detection, round robin, among others. The policy attributes **138** can include policy statistics such as a number of times a policy was enforced or a number of times a policy was not enforced. The policy attributes **138** can also include associations with network traffic data. For example, flows found to be non-conformant can be linked or tagged with corresponding policies to assist in the investigation of non-conformance.

**[0048]** The analytics engine **110** may include any number of engines **150**, including for example, a flow engine **152** for identifying flows (e.g., flow engine **152**) or an attacks engine **154** for identify attacks to the network. In some embodiments, the analytics engine can include a separate distributed denial of service (DDoS) attack engine **155** for specifically detecting DDoS attacks. In other embodiments, a DDoS attack engine may be a component or a sub-engine of a general attacks engine. In some embodiments, the attacks engine **154** and/or the DDoS engine **155** can use machine learning techniques to identify security threats to a network. For example, the attacks engine **154** and/or the DDoS engine **155** can be provided with examples of network states corresponding to an attack and network states corresponding to normal operation. The attacks engine **154** and/or the DDoS engine **155** can then analyze network traffic data to recognize when the network is under attack. In some embodiments, the network can operate within a trusted environment for a time to establish a baseline for normal network operation for the attacks engine **154** and/or the DDoS.

**[0049]** The analytics engine **110** may further include a search engine **156**. The search engine **156** may be configured, for example to perform a structured search, an NLP (Natural Language Processing) search, or a visual search. Data may be provided to the engines from one or more processing components.

**[0050]** The analytics engine **110** can also include a policy engine **158** that manages network policy, including creating and/or importing policies, monitoring policy conformance and non-conformance, enforcing policy, simulating changes to policy or network elements affecting policy, among other policy-related tasks.

**[0051]** ADM module **140** can determine dependencies of applications of the network. That is, particular patterns of traffic may correspond to an application, and the interconnectivity or dependencies of the application can be mapped to generate a graph for the application (i.e., an application dependency mapping). In this context, an application refers to a set of networking components that provides connectivity for a given set of workloads. For example, in a three-tier architecture for a web application, first endpoints of the web tier, second endpoints of the application tier, and third endpoints of the data tier make up the web application. ADM module **140** can receive input data from various repositories of the data lake **130** (e.g., the flow attributes **132**, the host and/or endpoint attributes **134**, the process attributes **136**, etc.). The ADM module **140** may analyze the input data to determine that there is first traffic flowing between external endpoints on port **80** of the first endpoints corresponding to Hypertext Transfer Protocol (HTTP) requests and responses. The input data may also indicate second traffic between first

ports of the first endpoints and second ports of the second endpoints corresponding to application server requests and responses and third traffic flowing between third ports of the second endpoints and fourth ports of the third endpoints corresponding to database requests and responses. The ADM module **140** may define an ADM for the web application as a three-tier application including a first EPG comprising the first endpoints, a second EPG comprising the second endpoints, and a third EPG comprising the third endpoints.

**[0052]** The presentation module **112** can include an application programming interface (API) or command line interface (CLI) **160**, a security information and event management (SIEM) interface **162**, and a web front-end **164**. As the analytics engine **110** processes network traffic and corresponding data and generates analytics data, the analytics data may not be in a human-readable form or it may be too voluminous for a user to navigate. The presentation module **112** can take the analytics data generated by analytics engine **110** and further summarize, filter, and organize the analytics data as well as create intuitive presentations for the analytics data.

**[0053]** In some embodiments, the API or CLI **160** can be implemented using Hadoop® Hive from Apache® for the back end, and Java® Database Connectivity (JDBC) from Oracle® Corporation of Redwood Shores, Calif., as an API layer. Hive is a data warehouse infrastructure that provides data summarization and ad hoc querying. Hive provides a mechanism to query data using a variation of structured query language (SQL) that is called HiveQL. JDBC is an application programming interface (API) for the programming language Java®, which defines how a client may access a database.

**[0054]** In some embodiments, SIEM interface **162** can be implemented using Kafka for the back end, and software provided by Splunk®, Inc. of San Francisco, Calif. as the SIEM platform. Kafka is a distributed messaging system that is partitioned and replicated. Kafka uses the concept of topics. Topics are feeds of messages in specific categories. In some embodiments, Kafka can take raw packet captures and telemetry information from the data mover **108** as input, and output messages to a SIEM platform, such as Splunk®. The Splunk® platform is utilized for searching, monitoring, and analyzing machine-generated data.

**[0055]** In some embodiments, web front-end **164** can be implemented using software provided by MongoDB®, Inc. of New York, N.Y. and Hadoop® ElasticSearch from Apache® for the back-end, and Ruby on Rails™ as the web application framework. MongoDB® is a document-oriented NoSQL database based on documents in the form of JavaScript® Object Notation (JSON) with dynamic schemas. ElasticSearch is a scalable and real-time search and analytics engine that provides domain-specific language (DSL) full querying based on JSON. Ruby on Rails™ is model-view-controller (MVC) framework that provides default structures for a database, a web service, and web pages. Ruby on Rails™ relies on web standards such as JSON or extensible markup language (XML) for data transfer, and hypertext markup language (HTML), cascading style sheets, (CSS), and JavaScript® for display and user interfacing.

**[0056]** Although FIG. **1** illustrates an example configuration of the various components of a network traffic monitoring system, those of skill in the art will understand that the components of the network traffic monitoring system **100** or any system described herein can be configured in a number of different ways and can include any other type and number of components. For example, the sensors **104**, the collectors **106**, the data mover **108**, and the data lake **130** can belong to one hardware and/or software module or multiple separate modules. Other modules can also be combined into fewer components and/or further divided into more components.

**[0057]** FIG. **2** illustrates an example of a network environment **200** in accordance with an embodiment. In some embodiments, a network traffic monitoring system, such as the network traffic monitoring system **100** of FIG. **1**, can be implemented in the network environment **200**. It should be understood that, for the network environment **200** and any environment discussed herein, there can be additional or fewer nodes, devices, links, networks, or components in similar or alternative configurations. Embodiments with different numbers and/or types of clients, networks, nodes, cloud components, servers, software components, devices, virtual or physical resources, configurations, topologies, services, appliances, deployments, or network devices are also contemplated herein. Further, the network environment **200** can include any number or type of resources, which can be accessed and utilized by clients or tenants. The illustrations and examples provided herein are for clarity and simplicity.

**[0058]** Network environment **200** can include network fabric **202**, a Layer 2 (L2) network **204**, a Layer 3 (L3) network **206**, and servers **208***a*, **208***b*, **208***c*, **208***d*, and **208***e* (collectively, **208**). The network fabric **202** can include spine switches **210***a*, **210***b*, **210***c*, and **210***d* (collectively, "**210**") and leaf switches **212***a*, **212***b*, **212***c*, **212***d*, and **212***e* (collectively, "**212**"). The spine switches **210** can connect to leaf switches **212** in the network fabric **202**. Leaf switches **212** can include access ports (or non-fabric ports) and fabric ports. The fabric ports can provide uplinks to the spine switches **210**, while the access ports can provide connectivity to endpoints (e.g., the servers **208**), internal networks (e.g., the L2 network **204**), or external networks (e.g., the L3 network **206**).

**[0059]** Leaf switches **212** can reside at the edge of the network fabric **202**, and can thus represent the physical network edge. For instance, in some embodiments, leaf switches **212***d* and **212***e* operate as border leaf switches in communication with edge devices **214** located in the external network **206**. Border leaf switches **212***d* and **212***e* may be used to connect any type of external network device, service (e.g., firewall, deep packet inspector, traffic monitor, load balancer, etc.), or network (e.g., the L3 network **206**) to the fabric **202**.

**[0060]** Although the network fabric **202** is illustrated and described herein as an example leaf-spine architecture, one of ordinary skill in the art will readily recognize that various embodiments can be implemented based on any network topology, including any data center or cloud network fabric. Indeed, other architectures, designs, infrastructures, and variations are contemplated herein. For example, the principles disclosed herein are applicable to topologies including three-tier (including core, aggregation, and access levels), fat tree, mesh, bus, hub and spoke, etc. Thus, in some embodiments, leaf switches **212** can be top-of-rack switches configured according to a top-of-rack architecture. In other embodiments, leaf switches **212** can be aggregation switches in any particular topology, such as end-of-row or middle-

of-row topologies. In some embodiments, leaf switches **212** can also be implemented using aggregation switches.

**[0061]** Moreover, the topology illustrated in FIG. **2** and described herein is readily scalable and may accommodate a large number of components, as well as more complicated arrangements and configurations. For example, the network may include any number of fabrics **202**, which may be geographically dispersed or located in the same geographic area. Thus, network nodes may be used in any suitable network topology, which may include any number of servers, virtual machines or containers, switches, routers, appliances, controllers, gateways, or other nodes interconnected to form a large and complex network. Nodes may be coupled to other nodes or networks through one or more interfaces employing any suitable wired or wireless connection, which provides a viable pathway for electronic communications.

**[0062]** Network communications in the network fabric **202** can flow through leaf switches **212**. In some embodiments, leaf switches **212** can provide endpoints (e.g., the servers **208**), internal networks (e.g., the L2 network **204**), or external networks (e.g., the L3 network **206**) access to the network fabric **202**, and can connect leaf switches **212** to each other. In some embodiments, leaf switches **212** can connect endpoint groups (EPGs) to the network fabric **202**, internal networks (e.g., the L2 network **204**), and/or any external networks (e.g., the L3 network **206**). EPGs are groupings of applications, or application components, and tiers for implementing forwarding and policy logic. EPGs can allow for separation of network policy, security, and forwarding from addressing by using logical application boundaries. EPGs can be used in the network environment **200** for mapping applications in the network. For example, EPGs can comprise a grouping of endpoints in the network indicating connectivity and policy for applications.

**[0063]** As discussed, the servers **208** can connect to the network fabric **202** via leaf switches **212**. For example, the servers **208***a* and **208***b* can connect directly to leaf switches **212***a* and **212***b*, which can connect the servers **208***a* and **208***b* to the network fabric **202** and/or any of the other leaf switches. Servers **208***c* and **208***d* can connect to leaf switches **212***b* and **212***c* via the L2 network **204**. Servers **208***c* and **208***d* and the L2 network **204** make up a local area network (LAN). LANs can connect nodes over dedicated private communications links located in the same general physical location, such as a building or campus.

**[0064]** The WAN **206** can connect to leaf switches **212***d* or **212***e* via the L3 network **206**. WANs can connect geographically dispersed nodes over long-distance communications links, such as common carrier telephone lines, optical light paths, synchronous optical networks (SONET), or synchronous digital hierarchy (SDH) links. LANs and WANs can include L2 and/or L3 networks and endpoints.

**[0065]** The Internet is an example of a WAN that connects disparate networks throughout the world, providing global communication between nodes on various networks. The nodes typically communicate over the network by exchanging discrete frames or packets of data according to predefined protocols, such as the Transmission Control Protocol/Internet Protocol (TCP/IP). In this context, a protocol can refer to a set of rules defining how the nodes interact with each other. Computer networks may be further interconnected by an intermediate network node, such as a router, to extend the effective size of each network. The endpoints **208** can include any communication device or component, such as a computer, server, blade, hypervisor, virtual machine, container, process (e.g., running on a virtual machine), switch, router, gateway, host, device, external network, etc.

**[0066]** In some embodiments, the network environment **200** also includes a network controller running on the host **208***a*. The network controller is implemented using the Application Policy Infrastructure Controller (APIC™) from Cisco®. The APIC™ provides a centralized point of automation and management, policy programming, application deployment, and health monitoring for the fabric **202**. In some embodiments, the APIC™ is operated as a replicated synchronized clustered controller. In other embodiments, other configurations or software-defined networking (SDN) platforms can be utilized for managing the fabric **202**.

**[0067]** In some embodiments, a physical server **208** may have instantiated thereon a hypervisor **216** for creating and running one or more virtual switches (not shown) and one or more virtual machines **218**, as shown for the host **208***b*. In other embodiments, physical servers may run a shared kernel for hosting containers. In yet other embodiments, the physical server **208** can run other software for supporting other virtual partitioning approaches. Networks in accordance with various embodiments may include any number of physical servers hosting any number of virtual machines, containers, or other virtual partitions. Hosts may also comprise blade/physical servers without virtual machines, containers, or other virtual partitions, such as the servers **208***a*, **208***c*, **208***d*, and **208***e*.

**[0068]** The network environment **200** can also integrate a network traffic monitoring system, such as the network traffic monitoring system **100** shown in FIG. **1**. For example, the network traffic monitoring system of FIG. **2** includes sensors **220***a*, **220***b*, **220***c*, and **220***d* (collectively, "**220**"), collectors **222**, and an analytics engine, such as the analytics engine **110** of FIG. **1**, executing on the server **208***e*. The analytics engine on server **208***e* can receive and process network traffic data collected by the collectors **222** and detected by the sensors **220** placed on nodes located throughout the network environment **200**. Although the analytics engine **208***e* is shown to be a standalone network appliance in FIG. **2**, it will be appreciated that analytics engine **208***e* can also be implemented as a virtual partition (e.g., VM or container) that can be distributed onto a host or cluster of hosts, software as a service (SaaS), or other suitable method of distribution. In some embodiments, the sensors **220** run on leaf switches **212** (e.g., the sensor **220***a*), hosts **208** (e.g., the sensor **220***b*), hypervisor **216** (e.g., the sensor **220***c*), and VMs **218** (e.g., the sensor **220***d*). In other embodiments, the sensors **220** can also run on the spine switches **210**, virtual switches, service appliances (e.g., firewall, deep packet inspector, traffic monitor, load balancer, etc.) and in between network elements. In some embodiments, sensors **220** can be located at each (or nearly every) network component to capture granular packet statistics and data at each hop of data transmission. In other embodiments, the sensors **220** may not be installed in all components or portions of the network (e.g., shared hosting environment in which customers have exclusive control of some virtual machines).

**[0069]** As shown in FIG. **2**, a host may include multiple sensors **220** running on the host (e.g., the host sensor **220***b*) and various components of the host (e.g., the hypervisor sensor **220***c* and the VM sensor **220***d*) so that all (or

substantially all) packets traversing the network environment **200** may be monitored. For example, if one of VMs **218** running on the host **208***b* receives a first packet from the WAN **206**, the first packet may pass through the border leaf switch **212***d*, the spine switch **210***b*, the leaf switch **212***b*, the host **208***b*, hypervisor **216**, and the VM. Since all or nearly all of these components contain a respective sensor, the first packet will likely be identified and reported to one of the collectors **222**. As another example, if a second packet is transmitted from one of VMs **218** running on the host **208***b* to the host **208***d*, sensors installed along the data path, such as at the VM **218**, hypervisor **216**, host **208***b*, leaf switch **212***b*, and host **208***d* will likely result in capture of metadata from the second packet.

**[0070]** FIG. **3** illustrates an example of a data pipeline **300** for generating network insights based on collected network information. The insights generated may include, for example, discovered applications or inventories, application dependencies, policies, efficiencies, resource and bandwidth usage, and network flows can be determined for the network using the network traffic data. In some embodiments, the data pipeline **300** can be directed by a network traffic monitoring system, such as the network traffic monitoring system **100** of FIG. **1**; an analytics engine, such as the analytics engine **110** of FIG. **1**; or other network service or network appliance. For example, an analytics engine **110** can be configured to discover of applications running in the network, map the applications' interdependencies, generate a set of proposed network policies for implementation, and monitor policy conformance and non-conformance among other network-related tasks.

**[0071]** The data pipeline **300** includes a data collection stage **302** in which network traffic data and corresponding data (e.g., host data, process data, user data, etc.) are captured by sensors (e.g., the sensors **104** of FIG. **1**) located throughout the network. The data may comprise, for example, raw flow data and raw process data. As discussed, the data can be captured from multiple perspectives to provide a comprehensive view of the network. The data collected may also include other types of information, such as tenant information, virtual partition information, out-of-band information, third party information, and other relevant information. In some embodiments, the flow data and associated data can be aggregated and summarized daily or according to another suitable increment of time, and flow vectors, process vectors, host vectors, and other feature vectors can be calculated during the data collection stage **302**. This can substantially reduce processing.

**[0072]** Data pipeline **300** can also include an input data stage **304** in which a network or security administrator or other authorized user may configure insight generation by selecting the date range of the flow data and associated data to analyze, and those nodes for which the administrator wants to analyze. In some embodiments, the administrator can also input side information, such as server load balance, route tags, and previously identified clusters during the input data stage **304**. In other aspects, the side information can be automatically pulled or another network element can push the side information.

**[0073]** The next stage of the data pipeline **300** is pre-processing **306**. During the pre-processing stage **306**, nodes of the network are partitioned into selected node and dependency node subnets. Selected nodes are those nodes for which the user requests application dependency maps and cluster information. Dependency nodes are those nodes that are not explicitly selected by the users for an ADM run but are nodes that communicate with the selected nodes. To obtain the partitioning information, edges of an application dependency map (i.e., flow data) and unprocessed feature vectors can be analyzed.

**[0074]** Other tasks can also be performed during the pre-processing stage **306**, including identifying dependencies of the selected nodes and the dependency nodes; replacing the dependency nodes with tags based on the dependency nodes' subnet names; extracting feature vectors for the selected nodes, such as by aggregating daily vectors across multiple days, calculating term frequency-inverse document frequency (tf-idf), and normalizing the vectors (e.g., $l_2$ normalization); and identifying existing clusters.

**[0075]** In some embodiments, the pre-processing stage **306** can include early feature fusion pre-processing. Early fusion is a fusion scheme in which features are combined into a single representation. Features may be derived from various domains (e.g., network, host, virtual partition, process, user, etc.), and a feature vector in an early fusion system may represent the concatenation of disparate feature types or domains.

**[0076]** Early fusion may be effective for features that are similar or have a similar structure (e.g., fields of TCP and UDP packets or flows). Such features may be characterized as being a same type or being within a same domain. Early fusion may be less effective for distant features or features of different types or domains (e.g., flow-based features versus process-based features). Thus, in some aspects, only features in the network domain (i.e., network traffic-based features, such as packet header information, number of packets for a flow, number of bytes for a flow, and similar data) may be analyzed. In other embodiments, analysis may be limited to features in the process domain (i.e., process-based features, such as process name, parent process, process owner, etc.). In yet other aspects, feature sets in other domains (e.g., the host domain, virtual partition domain, user domain, etc.) may be the.

**[0077]** After pre-processing, the data pipeline **300** may proceed to an insight generation stage **308**. During the insight generation stage **308**, the data collected and inputted into the data pipeline **300** may be used to generate various network insights. For example, an analytics engine **110** can be configured to discover of applications running in the network, map the applications' interdependencies, generate a set of proposed network policies for implementation, and monitor policy conformance and non-conformance among other network-related tasks. Various machine learning techniques can be implemented to analyze feature vectors within a single domain or across different domains to generate insights. Machine learning is an area of computer science in which the goal is to develop models using example observations (i.e., training data), that can be used to make predictions on new observations. The models or logic are not based on theory but are empirically based or data-driven.

**[0078]** After clusters are identified, the data pipeline **300** can include a post-processing stage **310**. The post-processing stage **310** can include tasks such as filtering insight data, converting the insight data into a consumable format, or any other preparations needed to prepare the insight data for consumption by an end user. At the output stage **312**, the generated insights may be provided to an end user. The end user may be, for example a network administrator, a third-

party computing system, a computing system in the network, or any other entity configured to receive the insight data. In some cases, the insight data may be configured to be displayed on a screen or provided to a system for further processing, consumption, or storage.

**[0079]** As noted above, a network traffic monitoring system may be configured to continually collect network data and generate various insights based on the collected network data. This network data and the insights may be updated over time and each set of network data and/or insights may provide a network snapshot or view of the state of the network for a particular period of time. The network snapshot may be generated periodically over time or in response to one or more events. Events may include, for example, a change to a network policy or configuration; an application experiencing latency that exceeds an application latency threshold; the network experiencing latency that exceeds a network latency threshold; failure of server, network device, or other network element; and similar circumstances. Various network snapshots may further be compared in order to identify changes in the state of the network over time and be used to provide additional insights into the operations of the network.

**[0080]** However, each network snapshot for an entire network or network cluster, may be quite large in size. The network may include a large number of nodes and a sensor may be implemented on some or all of the nodes in the network. Nodes may include, for example, a virtual partition (e.g., VM or container); a hypervisor or shared kernel managing one or more virtual partitions and/or physical servers, an application-specific integrated circuit (ASIC) of a switch, router, gateway, or other networking device, or a packet capture (pcap) appliance (e.g., a standalone packet monitor, a device connected to a network devices monitoring port, a device connected in series along a main trunk of a datacenter, or similar device), servers, end-user devices, or other element of a network. The amount of data generated by these sensors and the insights that may be derived from the data may be quite large. Furthermore, the network state may be updated often, causing several network snapshots to be created and created often.

**[0081]** There are many technical limitations to transmitting and receiving that amount of data often enough for end users to stay in sync, especially when multiple network snapshots are generated in relatively quick succession. Additionally, in some cases, the network snapshots may be requested by a large number of end users or consumers of the network snapshots. Communication of such large amounts of data to multiple entities may be difficult for the network traffic monitoring system to transmit and difficult for each entity to receive, especially when the network traffic monitoring system and/or each receiving entity may crash or deal with transmission errors. Aspects of the subject technology address these technical problems by fragmenting each network snapshot into smaller chunks or segments and using a stream processing service to publish the network snapshots to one or more subscribers.

**[0082]** FIG. **4** illustrates an example of a network traffic monitoring system **410** providing network snapshots to subscribers **430**, in accordance with various embodiments. An analytics engine **415** of a network monitoring system **410** may be configured to collect the network data generated by sensors deployed in a network, derive insights based on the network data, and generate network snapshots based on the network data and/or insights. The network snapshots may be generated periodically over time (e.g., every 15 minutes) and/or in response to network changes or events. Each network snapshot that is generated by the analytics engine **415** may be quite large in size. Accordingly, the analytics engine **415** may be configured to partition a network snapshot into smaller network snapshot segments (e.g., a 10 megabyte segment) and transmit the network snapshot segments to a stream processing service **420**.

**[0083]** According to some embodiment, the stream processing service **420** may be implemented in the network traffic monitoring system **410**, as part of the analytics engine **415**, or in the network managed by the network traffic monitoring system **410**. In other aspects, however, the stream processing service **420** may be outside of the network managed by the network traffic monitoring system **410**. The stream processing service **420** is configured to receive network snapshots or network snapshot segments from the analytics engine **415** and publish the network snapshot segments to one or more subscribers via a snapshot stream **425**.

**[0084]** According to other embodiments, the analytics engine **415** may provide the network snapshots to the stream processing service **420** and the stream processing service **420** may partition each snapshot into smaller network snapshot segments for the snapshot stream **425**.

**[0085]** A subscriber **430** may be a device, application, or other entity that can make use of the network snapshots provided by the analytics engine **415**. For example, the subscriber **430** may use the network snapshots to update network policies, generate additional insights, or store for analysis. The stream reader **435** may be a component of the subscriber **430** that subscribes to the snapshot stream **425** of the stream processing service **420**. The stream reader **435** is further configured to receive messages from the stream processing service **420** that include one or more network snapshot segments and reconstruct the network snapshot based on the received network snapshot segments. The reconstruction of a network snapshot faces additional technical obstacles in cases where subscribers **430** start receiving network snapshot segments in the middle of a network snapshot or the sequence of network snapshot segments that are received is broken.

**[0086]** Various aspects of the subject technology address these and other technical obstacles by having the analytics engine **415** annotate or tag each network snapshot segment with metadata that may be used by the stream reader **435** to reconstruct a network snapshot. The metadata may include, for example a "start" label, an "end" label, and/or a number associated with the location of the network snapshot segment within the network snapshot. For example, a first segment of a network snapshot may be tagged with the "start" label and a last segment of the network snapshot may be tagged with the "end" label. Each segment of a snapshot may also be numbered (e.g., from 0 to n, where n is assigned to the last segment). The network segments of the network snapshot are then provided to the stream processing service **420** for distribution in a snapshot output stream **425**.

**[0087]** Once the stream reader **435** subscribes to the snapshot stream **425** or otherwise begins receiving stream messages from the stream processing service **420**, the stream reader **435** may use the tags (e.g., start/end tags) and/or the enumeration of a received network snapshot segment to determine which network snapshot segment has been

received and which network snapshot segments, if any, should be requested from the stream processing service. For example, if the first network snapshot segment received by the stream reader **435** is segment 5, the stream reader **435** may transmit requests for segments 0-4 from the stream processing service **420** or request transmission of the stream starting from 5 segments back from the recently received network snapshot segment (e.g., rewinding the stream). Once a network snapshot segment tagged with the "end" label is received, the stream reader **435** may compile the network snapshot segments into a network snapshot.

**[0088]** FIG. **5** illustrates an example of a process **500** for compiling a network snapshot, in accordance with an embodiment. It should be understood that, for any process discussed herein, there can be additional, fewer, or alternative steps performed in similar or alternative orders, or in parallel, within the scope of the various embodiments unless otherwise stated. Process **500** can be performed by a computing device, and particularly, a subscriber device (e.g., the subscriber **430** of FIG. **4**), a stream reader application (e.g., the stream reader **435** of FIG. **4**), or similar system.

**[0089]** The subscriber device may subscribe to a snapshot stream provided by a stream processing service. The stream processing service may be configured to publish network snapshot segments to potentially a large number of subscribers. The network snapshot segments may be provided to subscribers in a series of messages that each include a network snapshot segment and metadata for the network snapshot segment (e.g., labels, tags, enumerations, segment identifiers, network snapshot identifiers, etc.) that may be used to indicate a position of the network snapshot segment within the snapshot stream.

**[0090]** Once the subscriber device is subscribed to the snapshot stream, at operation **505**, the subscriber device may receive a message containing a network snapshot segment from a stream processing service. Based on the metadata also contained in the message, the subscriber device may identify an offset associated with the network snapshot segment received in the message at operation **510**. Based on the offset, the subscriber device may determine whether the network snapshot segment is the first segment for the network snapshot or whether additional network snapshot segments should be requested at operation **515**.

**[0091]** For example, if the metadata for the network snapshot segment contains a "start" label or a "0" for the enumerated snapshot offset identifier, the subscriber device may determine that the offset for the network snapshot segment is 0, indicating that the network snapshot segment received in the message is the first network snapshot segment for a network snapshot. Accordingly, the subscriber device has not missed any network snapshot segments for the current network snapshot. The subscriber device can continue to operation **525** and receive subsequent network snapshot segments for the network snapshot from the snapshot stream and compile the network snapshot without having to request other network snapshot segments that the subscriber device has missed.

**[0092]** If, on the other hand, the metadata for the network snapshot segment does not contain the "start" label or the "0" value for the enumerated snapshot segment offset identifier, the subscriber device may determine that the network snapshot segment received in the message is not the first network snapshot segment for the network snapshot. If the subscriber device has not received the previous network snapshot segments for the network snapshot, the subscriber device may request additional messages containing the previous network snapshot segments from the stream processing service at operation **520**.

**[0093]** There may be several reasons for the subscriber device not having received one or more of the previous network snapshot segments. For example, the subscriber device may have just subscribed to the snapshot stream and the network snapshot segment received at operation **505** may be the first segment received for the network snapshot. The subscriber device or the stream reader associated with the subscriber device may have just restarted operation or one or more of the previous snapshot segments may not have been transmitted properly based on network failures, interface failures, or errors by the subscriber device or the stream processing service.

**[0094]** As an illustrative example, the metadata for the network snapshot segment may contain a snapshot offset identifier value of "8." The subscriber device may determine that the offset for the network snapshot segment is 8, indicating that there are 8 previous network snapshot segments (e.g., network snapshot segments with offset identifier values 0-7) for the current network snapshot before the current network snapshot. If the subscriber device has not received these previous network snapshot segments, the subscriber device may request them from the stream processing service at operation **520**. These network stream segments may be provided out-of-stream. That is, the stream processing service may provide them separately to the subscriber device upon request, outside the normal sequential operation of the snapshot stream.

**[0095]** While the requested network snapshot segments are being sent by the stream processing service and/or after they are received by the subscriber device, the subscriber device can continue receiving additional message from the snapshot stream at operation **525**. Once the last network snapshot segment for a network snapshot is received, the subscriber device may compile the network snapshot segments into a network snapshot at operation **530**. The subscriber device may identify the last network snapshot segment for the network snapshot because the metadata associated with the network snapshot segment may include an "end" or "last" label.

**[0096]** After the network snapshot segments for a first network snapshot are received and compiled, the subscriber device may continue to receive messages from the snapshot stream containing additional network snapshot segments for subsequent network snapshots. As a result, a series of network snapshots generated by a network traffic monitoring system may be provided to a stream processing service and provided to one or more subscribers in network snapshot segments that may be compiled by the subscriber system. These network snapshots may further processed or analyzed by the subscriber device for further use by the subscriber device or downstream consumers. For example, according to some embodiments, after a network snapshot is compiled, the subscriber device or the stream reader associated with the subscriber device may convert the network snapshot into a generic format that may be consumed by a group of diverse downstream consumers.

**[0097]** As noted above, the network snapshots may represent the state of a network or information associated with the network during a particular time period. Accordingly, multiple network snapshots that are compiled by the sub-

scriber system may be compared in order to identify changes in the state of the network over time and be used to provide additional insights into the operations of the network. The subscriber system may further be configured to compare two or more network snapshots, identify the differences between the network snapshots, and provide the differences to a downstream consumer.

**[0098]** FIG. **6** illustrates an example of a process **600** for identifying a difference between network snapshots, in accordance with an embodiment. It should be understood that, for any process discussed herein, there can be additional, fewer, or alternative steps performed in similar or alternative orders, or in parallel, within the scope of the various embodiments unless otherwise stated. Process **600** can be performed by a computing device, and particularly, a subscriber device (e.g., the subscriber **430** of FIG. **4**), a stream reader application (e.g., the stream reader **435** of FIG. **4**), or similar system.

**[0099]** At operation **605**, a stream reader or similar system may receive from a stream processing system a set of messages that contain network snapshot segments. The messages may be received based on process **500** of FIG. **5** or a similar process. At operation **610**, the stream reader may compile the network snapshot segments into two or more network snapshots. For the sake of illustrating the process of FIG. **6**, a first network snapshot and a second network snapshot may be compiled.

**[0100]** At operation **615**, the stream reader compares the compiled network snapshots to identify a difference between the first network snapshot and the second network snapshot. This difference may represent a change in a network policy or other data associated with the network. At operation **620**, the difference may then be provided to one or more consumer systems. In some cases, the difference may be converted into a generic format that is more accessible to the consumer systems. The consumer systems may be downstream consumers of network information and may be configured to use the difference to update network information stored by the consumer system rather than store the entire network snapshot. By using the difference to update the network information rather than an entire network snapshot, network resources (e.g., bandwidth), memory, processing time, and other computing resources may be conserved or more efficiently utilized.

**[0101]** According to some embodiments, downstream consumers systems of the network information may be interested in different portions of the network snapshots. For example, one consumer system may be interested in one set of endpoint groups in the network while another consumer system may be interested in another set of endpoint groups. These two sets of endpoint groups may or may not have overlapping portions. In order to reduce the information provided to the consumer system and further improve the efficient use of computing resources, the stream reader may filter out portions of the network snapshots or the identified differences between network snapshots provided to each consumer system based on filtering criteria associated with each consumer system. The filtering criteria may include, for example, endpoint group identifiers, a list of IP addresses of interest, a list of network policies of interest, or any other criteria that may be used to filter, categorize, or group network information or insights generated by the network traffic monitoring system.

**[0102]** FIG. **7** illustrates an example architecture **700** for implementing a secure policy update publishing stream, according to some aspects of the technology. Architecture **700** includes customer **701** that represents a party or network entity that wishes to subscribe to network policy updates, e.g., via a policy stream. As illustrated, customer **701** is communicatively coupled with a network monitoring appliance (e.g., Tetration backend **703**), as well as a processing pipeline **707**. Additionally, the Tetration backend **703** is coupled to a vendor/service provider **705**.

**[0103]** It is understood that the topology of architecture **700** is provided for illustrative purposes, and that additional monitoring appliances (e.g., messaging queues), vendors, or message queue pipelines may be implemented, without departing the scope of the disclosed technology. Additionally, vendor **705** can represent any third-party provider, for example, that is capable of providing software, or computing services over a computer network. By way of example, vendor **705** may represent a product provider such as Citrix, or F5 Networks, etc. Additionally, processing pipeline **707** can represent any messaging cueing service, such as that provided using Kafka for the back-end, and/or software provided by Splunk®, Inc., as the SIEM platform, as discussed above.

**[0104]** The architecture **700** illustrated in FIG. **7** provides systems configured for implementing a secure policy update process of the subject technology. In practice, customer **701** can instantiate a network monitoring appliance, i.e. Tetration backend **703**, by providing necessary commands and configuration parameters to a cloud based monitoring cluster. Once instantiated, Tetration backend **703** produces a customer certificate that is unique to customer **701**. The customer certificate can be generated based on configuration parameters and/or other identifying information about customer **701**.

**[0105]** Separately, a processing pipeline **707** can also be instantiated and connected to Tetration backend **703**. Processing pipeline **707** may be instantiated by customer **701**, or as an automatic process performed by Tetration backend **703**, for example, that is performed in response to instantiation of Tetration backend **703**. Once the customer certificate is generated for customer **701**, a second certificate corresponding to the customer certificate is generated, i.e. a Tetration certificate is generated identifying the relationship between Tetration backend **703** and customer **701**. The Tetration certificate is provided to processing pipeline **707**. Because the Tetration certificate corresponds with the generation of the customer certificate, the Tetration certificate can be used by processing pipeline to validate identity of customer **701**, for example, when customer **701** provides the customer certificate.

**[0106]** Subsequently, customer **701** can connect to processing pipeline **707** in order to subscribe to policy updates that are pushed by Tetration backend **703**. For example, as illustrated in FIG. **7**, a policy update provided by Tetration backend **703** to processing pipeline **707** can be received by customer **701**, after customer **701** provides the customer certificate to processing pipeline **707** (e.g., after authenticating his identity). In some aspects, the policy stream provided by processing pipeline **707** can be encrypted. As such, potentially malicious users would not only need to spoof both the Tetration and customer certificates, but would also need to be able to decrypt any policy updates provided on the policy stream. These multiple layers of security

provide added safeguards against man-in-the-middle type attacks i.e., where malevolent users/hackers may attempt to learn policy information being provided by the Tetration backend **703**.

**[0107]** In some aspects, additional information contained in a vendor certificate may be used for customer/subscriber authentication. For example, vendor **705** can provide a vendor certificate to Tetration backend **703** and/or to the processing pipeline **707**. The vendor certificate may be further used to authenticate the customer certificate that is generated by Tetration backend **703**.

**[0108]** FIG. **8** illustrates an example process **800** for authenticating customer access to a secure policy stream, e.g., for receiving network policy updates, according to some aspects of the technology. Process **800** begins when a network monitoring device is instantiated in response to request (e.g., from a subscriber/customer), wherein the request includes one or more parameters for the network monitoring device (**802**). As discussed above, the request can be provided by customer e.g. via a network connected device, such as discussed above with respect to customer **701**. Additionally the network monitoring device can be a monitoring appliance that includes one or more computing clusters, such as a Tetration cluster, e.g. Tetration backend **703**.

**[0109]** Subsequently, a first certificate (e.g. customer certificate) is received from the network monitoring device, wherein the first certificate is based on the configuration parameters (**804**). As discussed above, the first certificate can be used to uniquely identify the subscriber/customer that instantiated the network monitoring application (e.g. Tetration cluster).

**[0110]** Once the subscriber/customer certificate has been received, or certificate (e.g. the first certificate) can be sent to a processing pipeline for authentication, wherein the processing pipeline is configured to authenticate the first certificate based on a second certificate that is received by the processing pipeline (**806**).

**[0111]** The processing pipeline may be implemented using a Directed Acyclic Graph buffer network, such as that implemented by a Kafka messaging system. As discussed above, the processing pipeline can also be configured to receive multiple certificates that can be used to validate/authenticate subscriber access to a policy update stream. In the current example, the second certificate can be a Tetration certificate that is generated by the Tetration cluster upon instantiation in step (**802**), discussed above. That is, the second certificate can be used by the Kafka messaging system to validate/authenticate identity of the subscriber/customer providing the first certificate (e.g. the customer certificate) to the messaging queue.

**[0112]** Subsequently, if the pipeline successfully authenticates the first certificate the subscriber/customer is provided access to one or more policy updates that are provided via policy stream provided by the processing pipeline (**808**). In this scenario, further to the above example discussed with respect to FIG. **7**, once properly authenticated using a customer certificate, customer **701** can be subscribed to a policy stream that is provided by processing pipeline **707** based on certificate validations performed for a Tetration certificate. In some aspects policy subscriptions and/or enforcements can be further authenticated using a vendor certificate, for example, that is provided in the code of one or more products provided by a vendor, e.g. vendor **705**, discussed above.

**[0113]** FIG. **9** illustrates an example of an electronic system with which some aspects of the subject technology can be implemented. Specifically, FIG. **9** illustrates an example network device **900**, which could include, but is not limited to a mobile device, such as a smart phone, a notebook computer, or a tablet computing device.

**[0114]** Network device **900** includes a master central processing unit (CPU) **962**, interfaces **968**, and a bus **915** (e.g., a PCI bus). When acting under the control of appropriate software or firmware, the CPU **962** is responsible for executing packet management, error detection, and/or routing functions. The CPU **962** preferably accomplishes all these functions under the control of software including an operating system and any appropriate applications software. CPU **962** can include one or more processors **963** such as a processor from the Motorola family of microprocessors or the MIPS family of microprocessors. In an alternative embodiment, processor **963** is specially designed hardware for controlling the operations of network device **900**. In a specific embodiment, a memory **961** (such as non-volatile RAM and/or ROM) also forms part of CPU **962**. However, there are many different ways in which memory could be coupled to the system.

**[0115]** The interfaces **968** can be provided as interface cards (sometimes referred to as "line cards"). Generally, they control the sending and receiving of data packets over the network and sometimes support other peripherals used with a router. Among the interfaces that can be provided are Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, and the like. In addition, various very high-speed interfaces can be provided such as fast token ring interfaces, wireless interfaces, Ethernet interfaces, Gigabit Ethernet interfaces, ATM interfaces, HSSI interfaces, POS interfaces, FDDI interfaces and the like. Generally, these interfaces may include ports appropriate for communication with the appropriate media. In some cases, they may also include an independent processor and, in some instances, volatile RAM. The independent processors may control such communications intensive tasks as packet switching, media control and management. By providing separate processors for the communications intensive tasks, these interfaces allow the master microprocessor **962** to efficiently perform routing computations, network diagnostics, security functions, etc.

**[0116]** Although the system shown in FIG. **9** is one specific network device of the present technology, it is by no means the only network device architecture on which the present technology can be implemented. For example, an architecture having a single processor that handles communications as well as routing computations, etc. is often used. Further, other types of interfaces and media could also be used with the router.

**[0117]** Regardless of the network device's configuration, it may employ one or more memories or memory modules (including memory **961**) configured to store program instructions for the general-purpose network operations and mechanisms for roaming, route optimization and routing functions described herein. The program instructions may control the operation of an operating system and/or one or more applications, for example. The memory or memories

may also be configured to store tables such as mobility binding, registration, and association tables, etc.

**[0118]** FIG. **10** illustrates a system bus computing system architecture **1000** wherein the components of the system are in electrical communication with each other using a bus **1005**. Exemplary system **1000** includes a processing unit (CPU or processor) **1010** and a system bus **1005** that couples various system components including the system memory **1015**, such as read only memory (ROM) **1020** and random access memory (RAM) **1025**, to the processor **1010**.

**[0119]** System **1000** can include a cache of high-speed memory connected directly with, in close proximity to, or integrated as part of the processor **1010**. The system **1000** can copy data from the memory **1015** and/or the storage device **1030** to the cache **1012** for quick access by the processor **1010**. In this way, the cache can provide a performance boost that avoids processor **1010** delays while waiting for data. These and other modules can control or be configured to control the processor **1010** to perform various actions. Other system memory **1015** can be available for use as well. Memory **1015** can include multiple different types of memory with different performance characteristics. The processor **1010** can include any general purpose processor and a hardware module or software module, such as module 1 **1032**, module 2 **1034**, and module 3 **1036** stored in storage device **1030**, configured to control the processor **1010** as well as a special-purpose processor where software instructions are incorporated into the actual processor design. The processor **1010** may essentially be a completely self-contained computing system, containing multiple cores or processors, a bus, memory controller, cache, etc. A multi-core processor can be symmetric or asymmetric.

**[0120]** To enable user interaction with the computing device **1000**, an input device **1045** can represent any number of input mechanisms, such as a microphone for speech, a touch-sensitive screen for gesture or graphical input, keyboard, mouse, motion input, speech and so forth. An output device **1035** can also be one or more of a number of output mechanisms known to those of skill in the art. In some instances, multimodal systems can enable a user to provide multiple types of input to communicate with the computing device **1000**. The communications interface **1040** can generally govern and manage the user input and system output. There is no restriction on operating on any particular hardware arrangement and therefore the basic features here may easily be substituted for improved hardware or firmware arrangements as they are developed.

**[0121]** Storage device **1030** is a non-volatile memory and can be a hard disk or other types of computer readable media which can store data that are accessible by a computer, such as magnetic cassettes, flash memory cards, solid state memory devices, digital versatile disks, cartridges, random access memories (RAMs) **1025**, read only memory (ROM) **1020**, and hybrids thereof.

**[0122]** The storage device **1030** can include software modules **1032**, **1034**, **1036** for controlling processor **1010**. Other hardware or software modules are contemplated. The storage device **1030** can be connected to the system bus **1005**. In one aspect, a hardware module that performs a particular function can include the software component stored in a computer-readable medium in connection with the necessary hardware components, such as the processor **1010**, bus **1005**, display **1035**, and so forth, to carry out the function.

**[0123]** By way of example, software modules **1032**, **1034**, **1036** can be configured for facilitating a multi-certificate authentication technique of the disclosed technology. For example, the software modules can be configured for performing steps to instantiate a network monitoring device in response to a request, the request comprising one or more configuration parameters for the network monitoring device, receive a first certificate from the network monitoring device, wherein the first certificate is based on the one or more configuration parameters, and send the first certificate to a processing pipeline for authentication, wherein the processing pipeline is configured to authenticate the first certificate based on a second certificate received by the processing pipeline from the network monitoring device. In some aspects, the modules can be further configured to perform steps for receiving one or more policy updates from a policy stream provided by the processing pipeline if the processing pipeline successfully authenticates the first certificate, decrypting the one or more policy updates received from the policy stream; and implementing at least one change indicated by the one or more policy updates to an associated customer network.

**[0124]** In some implementations, the processing pipeline includes multiple buffers arranged in a Directed Acyclic Graph (DAG) configuration. Additionally, in some implementations, the processing pipeline includes a Kafka distributed messaging system.

**[0125]** For clarity of explanation, in some instances the various embodiments may be presented as including individual functional blocks including functional blocks comprising devices, device components, steps or routines in a method embodied in software, or combinations of hardware and software.

**[0126]** In some embodiments the computer-readable storage devices, mediums, and memories can include a cable or wireless signal containing a bit stream and the like. However, when mentioned, non-transitory computer-readable storage media expressly exclude media such as energy, carrier signals, electromagnetic waves, and signals per se.

**[0127]** Methods according to the above-described examples can be implemented using computer-executable instructions that are stored or otherwise available from computer readable media. Such instructions can comprise, for example, instructions and data which cause or otherwise configure a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Portions of computer resources used can be accessible over a network. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, firmware, or source code. Examples of computer-readable media that may be used to store instructions, information used, and/or information created during methods according to described examples include magnetic or optical disks, flash memory, USB devices provided with non-volatile memory, networked storage devices, and so on.

**[0128]** Devices implementing methods according to these disclosures can comprise hardware, firmware, and/or software, and can take any of a variety of form factors. Typical examples of such form factors include laptops, smart phones, small form factor personal computers, personal digital assistants, rackmount devices, standalone devices, and so on. Functionality described herein also can be embodied in peripherals or add-in cards. Such functionality

can also be implemented on a circuit board among different chips or different processes executing in a single device, by way of further example.

**[0129]** The instructions, media for conveying such instructions, computing resources for executing them, and other structures for supporting such computing resources are means for providing the functions described in these disclosures.

**[0130]** Although a variety of examples and other information was used to explain aspects within the scope of the appended claims, no limitation of the claims should be implied based on particular features or arrangements in such examples, as one of ordinary skill would be able to use these examples to derive a wide variety of implementations. Further and although some subject matter may have been described in language specific to examples of structural features and/or method steps, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to these described features or acts. For example, such functionality can be distributed differently or performed in components other than those identified herein. Rather, the described features and steps are disclosed as examples of components of systems and methods within the scope of the appended claims.

**1**. A computer-implemented method for authenticating a subscriber to a policy stream, comprising:
- instantiating a network monitoring device in response to a request, the request comprising one or more configuration parameters for the network monitoring device;
- receiving a first certificate from the network monitoring device, wherein the first certificate is based on the one or more configuration parameters; and
- sending the first certificate to a processing pipeline for authentication, wherein the processing pipeline is configured to authenticate the first certificate based on a second certificate received by the processing pipeline from the network monitoring device.

**2**. The computer-implemented method of claim **1**, further comprising:
- receiving one or more policy updates from a policy stream provided by the processing pipeline if the processing pipeline successfully authenticates the first certificate.

**3**. The computer-implemented method of claim **2**, further comprising:
- decrypting the one or more policy updates received from the policy stream; and
- implementing at least one change indicated by the one or more policy updates to an associated customer network.

**4**. The computer-implemented method of claim **1**, wherein the processing pipeline comprises a plurality of buffers arranged in a Directed Acyclic Graph (DAG) configuration.

**5**. The computer-implemented method of claim **1**, wherein the processing pipeline comprises a Kafka distributed messaging system.

**6**. The computer-implemented method of claim **1**, wherein the processing pipeline is further configured to authenticate the first certificate based on a third certificate provided by a third-party vendor.

**7**. The computer-implemented method of claim **6**, wherein the third certificate is integrated into product code in a software package provided by the third-party vendor.

**8**. A non-transitory computer-readable medium having computer readable instructions that, upon being executed by a processor, cause the processor to:
- instantiate a network monitoring device in response to a request, the request comprising one or more configuration parameters for the network monitoring device;
- receive a first certificate from the network monitoring device, wherein the first certificate is based on the one or more configuration parameters; and
- send the first certificate to a processing pipeline for authentication, wherein the processing pipeline is configured to authenticate the first certificate based on a second certificate received by the processing pipeline from the network monitoring device.

**9**. The non-transitory computer-readable medium of claim **8**, wherein the instructions are further configured to cause to processor to:
- receive one or more policy updates from a policy stream provided by the processing pipeline if the processing pipeline successfully authenticates the first certificate.

**10**. The non-transitory computer-readable medium of claim **9**, wherein the instructions are further configured to cause to processor to:
- decrypting the one or more policy updates received from the policy stream; and
- implementing at least one change indicated by the one or more policy updates to an associated customer network.

**11**. The non-transitory computer-readable medium of claim **8**, wherein the processing pipeline comprises a plurality of buffers arranged in a Directed Acyclic Graph (DAG) configuration.

**12**. The non-transitory computer-readable medium of claim **8**, wherein the processing pipeline comprises a Kafka distributed messaging system.

**13**. The non-transitory computer-readable medium of claim **8**, wherein the processing pipeline is further configured to authenticate the first certificate based on a third certificate provided by a third-party vendor.

**14**. The non-transitory computer-readable medium of claim **13**, wherein the third certificate is integrated into product code in a software package provided by the third-party vendor.

**15**. A system comprising:
- a processor; and
- memory including instructions that, upon being executed by the processor, cause the system to:
- instantiate a network monitoring device in response to a request, the request comprising one or more configuration parameters for the network monitoring device;
- receive a first certificate from the network monitoring device, wherein the first certificate is based on the one or more configuration parameters; and
- send the first certificate to a processing pipeline for authentication, wherein the processing pipeline is configured to authenticate the first certificate based on a second certificate received by the processing pipeline from the network monitoring device.

**16**. The system of claim **15**, wherein the instructions are further configured to cause to processor to:
- receive one or more policy updates from a policy stream provided by the processing pipeline if the processing pipeline successfully authenticates the first certificate.

**17**. The system of claim **16**, wherein the instructions are further configured to cause to processor to:

decrypting the one or more policy updates received from the policy stream; and

implementing at least one change indicated by the one or more policy updates to an associated customer network.

**18**. The system of claim **15**, wherein the processing pipeline comprises a plurality of buffers arranged in a Directed Acyclic Graph (DAG) configuration.

**19**. The system of claim **15**, wherein the processing pipeline comprises a Kafka messaging system.

**20**. The system of claim **15**, wherein the processing pipeline is further configured to authenticate the first certificate based on a third certificate provided by a third-party vendor.

* * * * *