Dumping network diagnostics (netdump)

Motivation

In this article we explained how EVE is able to hook into the HTTP client to monitor any request made towards the controller or an image datastore, and collect a so called *network trace* - a summary of what was happening behind the scenes during HTTP request processing.

Network tracing comes with an additional overhead and the output can be quite large (JSON with tens of kilobytes in size). Therefore, we must be careful about how often are network traces obtained and how do we publish them. For example, logging network trace as a single message is not an option. Instead, EVE publishes network traces inside Tar/GZip archives, labeled as "***netdumps***", by storing them persistently under `/persist/netdump` directory (for now EVE does not upload them to the cloud or anywhere else remote). This is done by the pillar's netdump package, which additionally adds some more files into each archive to capture the config/state of the device connectivity at the moment of the publication. All this information combined allows to troubleshoot a connectivity issue (between device and the controller or a data-store) even after it is no longer reproducible. Ideally, it should not be required to ask a customer for more (networking-specific) information to better understand the issue, let alone to run some commands and retrieve the output for us (because this has already been done automatically by netdump).

Netdump

Every published netdump package contains:

- *eve directory* with snapshots of pubsub publications related to device connectivity (such as `DevicePortConfigList`, `DeviceNetworkStatus`, `AssignableAdapters`), DOT files projecting the intended and the current network config as dependency graphs, wwan config/status/metrics and also files with basic info like the EVE and Golang versions. More info on these files can be found inside the EVE documentation.
- *`linux` directory* with the output from networking-related commands like `ip`, `arp`, `iptables`, `netstat`, etc., all executed at the time of the publication.
- *`requests` directory* with attached network traces and packet captures (if enabled) collected for HTTP requests run by EVE. This can be for example request to get device config or just to ping the controller, or to download an image. Note that EVE can make multiple attempts for every request, trying every port and local IP address. Every attempt will have a separate trace + pcaps under its own subdirectory named: `<request-name>-[<interface>-]<attempt-index>`

Every netdump is published into a ***topic***, represented by its name and by default limited in size to 10 netdumps at most (configurable by `netdump.topic.maxcount`). The oldest netdump of a topic is unpublished (removed from `/persist/netdump`) should a new netdump exceed the limit. Topics are used to separate different microservices and even to split successful and failed requests from each other. Topic name is therefore typically: `<microservice>-<ok|fail>`. For troubleshooting purposes, netdumps of failed requests are obviously more useful, but having a trace of a "good run" can be used to compare with a "bad run" and find differences. Published netdump filename is a concatenation of the topic name with a publication timestamp plus the `.tgz` extension, so for example: `downloader-fail-2023-01-03T14-25-04.tgz`, `nim-ok-2023-01-03T13-30-36`, etc.

Sources of Netdumps

Not all microservices that communicate over the network are traced and contribute with netdumps. Currently traced HTTP requests are:

- `/ping` request done by nim to verify connectivity for the *latest* DPC (testing of older DPCs is never traced). Packet capture is also enabled and the obtained pcap files are included in the published netdumps. To limit the overhead associated with tracing and packet capture, nim is only allowed to enable them and produce netdump at most once per day (configurable by `netdump.topic.postonboard.interval`). However, before device is fully onboarded a much lower interval configured by `netdump.topic.preonboard.interval` (by default one hour) is applied to get more frequent diagnostics for initial connectivity troubleshooting.
- `/config` and `/info` requests done by zedagent to obtain device configuration and publish info messages, respectively. Packet capture is not enabled in this case. Follows the same interval as given by `netdump.topic.postonboard.interval`. For `/info` requests, tracing only covers publication of the `ZInfoDevice` message. Moreover, tracing is enabled only if the highest priority DPC is currently being applied and is reported by nim as working. Otherwise, we will eventually get `nim-fail*` netdump which should be sufficient for connectivity troubleshooting. The purpose of zedagent netdumps is to debug issues specific to `/config` an `/info` requests (which are essential to keep the device remotely manageable). Netdumps are published separately into topics `zedagent-config-<ok|fail>` and `zedagent-info-<ok|fail>`.
- *every* download request performed by downloader using the HTTP protocol is traced and netdump is published into the topic `downloader-ok` or `downloader-fail`, depending on the outcome of the download process. By default, this does not include packet captures. However, a limited PCAP can be enabled with config option `netdump.downloader.with.pcap`. Limited in the sense that it will not include TCP segments carrying non-empty payload (i.e. packets with the downloaded data). The total PCAP size is also limited to 64MB (packets past this limit will not be included).

Retrieving Netdumps

In order to troubleshoot a present or a past connectivity issue, it is necessary to locate and obtain the appropriate netdump from the affected device - locate by microservice aka topic name and look for the closest timestamp. Without a remote connectivity to the device, it is possible to dump all diagnostics to a USB stick. See CONFIG.md, section "Creating USB sticks". With this method, the entire `/persist/netdump` directory will be copied over. If device is remotely accessible, published netdumps can be listed and copied over ssh (if enabled by config), edgeview (`ls` and `cp` commands) or using a remote console if available.