Inspur NOS 系统架构白皮书

文档版本 V1.0

发布日期 2022-12-16

商标说明

Inspur 浪潮、Inspur、浪潮、Inspur NOS 是浪潮集团有限公司的注册商标。

本手册中提及的其他所有商标或注册商标，由各自的所有人拥有。

技术支持

技术服务电话：400-860-0011

地　　址：中国济南市浪潮路 1036 号

浪潮电子信息产业股份有限公司

邮　　箱：lckf@inspur.com

邮　　编：250101

前言

变更记录

版本	时间	变更内容
V1.0	2022-12-16	首版发布

目 录

1 概述

1.1 背景

早在 SONiC 出现以前，HP 在 2015 年就建立了 OpenSwitch 社区，并发布开源操作系统 OPS，与 Accton，Arista，Broadcom，Intel 和 VMware，一起推进基于该社区的开放网络平台发展。虽然该平台赋予了开发者和用户快速创新的能力，使客户能够快速构建数据中心网络，但由于不是最终用户主导，缺少用户支持，导致日渐式微，目前已经无人问津。而 2017 发布的 SONiC 系统的主导者是微软、阿里、腾讯这些大规模公有云厂商。有了客户的强烈追捧，芯片厂商、OEM 设备商和软件开发商都有强劲的动力参与社区，从而使社区日渐繁荣。

SONiC 在系统架构设计上吸取了商业 NOS 的经验和开源社区的最新成果，在整体架构上引入了中央数据库，各个模块组件都以 docker 方式发布，而且只和中央数据库交互。这种设计使得各个模块松耦合，基于 docker 的模块非常容易从其他开源社区导入代码，非常适用于社区开发，也为用户快速创新提供了最大的便利。

SONiC 是基于 Linux 的开源网络操作系统，其优点是可部署在不同芯片和不同硬件的交换机上运行。它是一个将传统交换机操作系统软件分解成多个容器化组件的创新方案，这使得增加新的组件和功能变得非常方便。由于 SONiC 的网络应用都是基于容器构建的，所以 SONiC 可以非常方便的在运行环境中实现不停机部署或升级应用。此外，SONiC 大量使用了现有的开源项目和开源技术，如 Docker、Redis、FRR、Teamd、LLDPD 以及自动化配置工具 Ansible、Puppet 和 Chef 等，这使得 SONiC 团队可以灵活地创建所需的网络解决方案，同时可以有效地利用大型生态系统和社区的集体力量来快速提供新的服务。目前，一些大型的云服务提供商已经将 SONiC 部署在其数据中心。

1.2 定义

Inspur NOS 是浪潮基于 SONiC 开发出的白盒交换机 Network OS。主要有两个重点实现：

1. 完善 SONiC 既有功能。
2. 基于客户的需求和未来趋势，提供浪潮自研的进阶网络功能与管理功能。

    - ACL：提供多种 ACL 规则类型，包括 L2、L3、IPv6、PFCWD、Mirror。
    - 控制平面 ACL：控制到 CPU 和上层应用的流量。

- 端口通道：提供静态模式并在运行时应用配置。
- 静态路由：当接口启动时应用静态路由，当接口删除后将其禁用。
- Config checker：一个集中式的配置检查库。可参阅 3.3.1 章节。
- inCLI commands：涵盖全功能的 Cisco-like CLI commands。可参阅 3.3.2 章节。
- 远程连线管理服务：提供远程连线管理进程的可配置性。可参阅 3.3.2 章节。
- Functional-driven Restful API：涵盖全功能的 Functional-driven Restful API。可参阅 3.3.4 章节。
- 告警机制：提供弹性可注册性的告警功能，可记录日志和透过 grpc 发出告警。可参阅 3.3.5 章节。
- 风暴控制：控制广播、组播、未知单播流量，以防止风暴影响网络运行。可参阅 3.3.6 章节。
- 防止环路：具有检测和避免环路的有效机制。可参阅 3.3.7 章节。
- License：浪潮提供了一个授权机制。已满足多样的商业场景需求搭配。可参阅 3.3.8 章节。
- IP 冲突辨别：避免服务器或是虚拟机可能会因为获取或被设定成相同的 IP 地址而产生冲突。可参阅 3.3.9 章节。
- TAM：带内网络遥测（INT）和丢包监控（MOD）。可参阅 3.3.10 章节。
- 侦测 MAC Flapping：具有检测和避免 mac 漂移的有效机制。可参阅 3.3.11 章节。
- 黑洞路由：使用了出接口为 NULL0 的路由，那么这些报文将被直接丢弃，因此出接口为 null0 的路由被称为黑洞路由。可参阅 3.3.12 章节。
- L2/L3 EVPN VXLAN：允许同一子网中的主机设备相互发送桥接或第 2 层流量。网络使用第 2 层虚拟网络实例（VNI）转发桥接流量。可参阅 3.3.13、3.3.14 章节。
- 策略路由：提供一种决定路由的方式，由网络管理者决定路由规则，再根据这些规

则来制定路由。可参阅 3.3.15 章节。

- NetConf：基于可扩展标记语言 XML 的网络配置和管理协议，使用简单的基于 RPC 机制实现客户端和服务器之间通信。可参阅 3.3.16 章节。
- sflow：提供取样的方式，获得网络报文的信息。可参阅 3.3.17 章节。
- ARP/ND 转主机路由：从接口（Ethernet/VLAN/Port-Channel）学到的 arp/nd 表项转换成 host route。可参阅 3.3.18 章节。
- L2 L3 interface definition：重新定义接口状态为四种状态，none-L2，L2，none-L3，L3。可参阅 3.3.19 章节。
- inTech-Support：提供一键收集功能，包含固件、软件等配置文件，也支持远程备份。可参阅 3.3.20 章节。
- 增强 CoPP 动态配置：原生 SONiC 社区提出的 CoPP 是仅提供初始配置档案，不允许用户修改配置。浪潮增强 CoPP 动态配置功能。可参阅 3.3.21 章节。
- 增加 ECMP HASH 动态配置：原生 SONiC 社区提出的方案仅提供哈希因子的初始配置档案，不允许用户修改配置。浪潮增强 ECMP HASH 动态配置。可参阅 3.3.22 章节。
- 可视化呈现：交换机上提供节点导出器使能测量各种机器资源，并上传到采集器，达到可视化接口呈现。可参阅 3.3.23 章节。
- DHCP Smart Relay：当没有来自 DHCP 服务器的 DHCP-OFFER 消息时，允许 DHCP 中继代理将网关地址切换到辅助地址。可参阅 3.3.24 章节。
- 动态负载平衡：提供基于 LAG 以及 ECMP 的 DLB 功能。可参阅 3.3.25 章节。
- inLog：配置每个程序的日志，时间区间查询、特定程序查询等功能。可参阅 3.3.26 章节。

- MC-LAG：支持跨设备链路聚合组。将两台交换机与被接入设备进行链路聚合协商，从被接入设备来看，对端的两台交换机被虚拟成了一台交换机。可参阅 3.3.27 章节。
- In-band ZTP：支持带内 ZTP 功能，方便客户环境批量布署。可参阅 3.3.28 章节。

后续章节会介绍 Inspur NOS 的软件架构与支持的协议。

2 缩写和术语

缩写和术语	解释
SONiC	Software for Open Networking in the Cloud
ACL	Access Control List
LPD	Loop Prevention Deamon
TAM	Telemetry and Monitoring
EVPN	Ethernet VPN（以太网络VPN）
VxLAN	Virtual Extensible LAN（虚拟局域网扩展）
PBR	Policy-Base Route（策略路由）
Netconf	Network Configuration Protocol
sFlow	sampled Flow
CoPP	Control Plane Policing
ECMP	Equal Cost Multi Path Routing
DHCP	Dynamic Host Configuration Protocol
DLB	Dynamic Load Balance
MC-LAG	Multi-Chassis Link Aggregation Group
ZTP	Zero Touch Provisioning（零配置部署）

3 技术介绍

本章主要介绍 InspurNOS 的技术特点，包含目前 SONiC 社群的使用方式和浪潮自有 NOS 的改进增强部分。

3.1 系统架构

图 3-1 Inspur NOS 架构

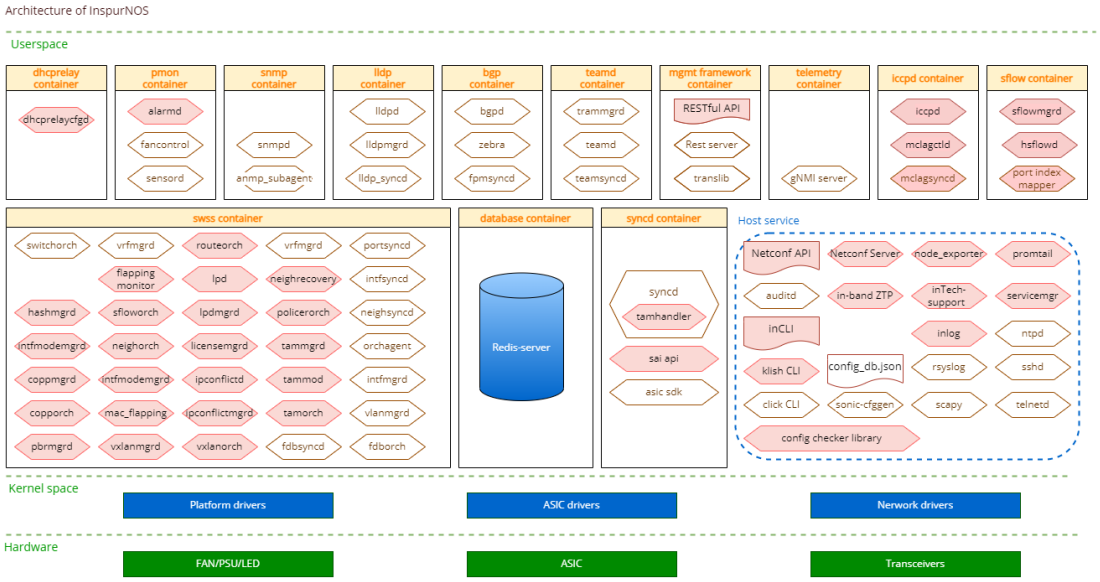


图 3-1 是 Inspur NOS 的架构图。Inspur 不仅完善了 SONiC 的功能，还开发了许多客户场景的需求功能。图 3-1 中的粉红色部分就是 Inspur 自研开发出的功能服务。

3.2 SONiC 社群原生系统

SONiC 系统的架构由各种模块组成，这些模块通过集中式和可扩展的基础架构相互交互。这个基础设施依赖于一个 Redis-DB 引擎：一个键值数据库来提供一个独立于语言的接口，一个在所有 SONiC 子系统之间进行数据持久化、复制和多进程通信的方法。

通过依赖 redis 引擎基础设施提供的发布者/订阅者消息传递范式，应用程序可以只订阅它们需要的数据视图，不会涉及与其功能无关的实现细节。

SONiC 将每个模块放置在独立的 Docker 容器中，以保持容器内所负责功能与库之间的高内聚性，减少脱节组件之间的耦合。每个组件都完全独立于较低层抽象交互所需的特定平台的细节。

SONiC 将其主要功能组件分解为以下 Docker 容器：

- bgp
- database
- dhcp-relay
- iccpd
- lldp
- mgmt-framework
- pmon
- radv
- sflow
- snmp
- swss
- syncd
- teamd
- telemetry

后续子章节会描述每个 Docker 容器中包含的功能，以及 Linux 主机系统运行的关键 SONiC 组件。

3.2.1 bgp container

运行支持的路由堆栈，FRR。尽管容器以正在使用的路由协议（BGP）命名，但实际上这些路由堆栈可以运行各种其他协议（例如 OSPF、ISIS、LDP 等）。

bgp container 功能细分如下：

- bgpd：常规 BGP 实现。来自外部各方的路由状态通过常规 TCP/UDP socket 接收，并通过 zebra/fpmsyncd 接口向下推送到转发平面。
- zebra：作为传统的 IP 路由管理器，它提供跨不同协议的内核路由表更新、接口查找和路由重新分配服务。Zebra 还负责将计算出的 FIB 向下推送到内核（通过 netlink 接口）和转发过程中涉及的南向组件（通过 Forwarding-Plane-Manager 接口--FPM--）。
- fpmsyncd：小型守护进程，负责收集 zebra 生成的 FIB 状态并将其内容转储到 Redis-engine 内的 Application-DB 表（APPL_DB）中。

3.2.2 database container

托管 Redis 数据库引擎。SONiC 应用程序可以通过 UNIX socket 访问此引擎中保存的数据库。这些是 Redis 引擎托管的主要数据库：

- APPL_DB：存储所有应用程序容器生成的状态——路由、下一跳、邻居等。这是所有希望与其他 SONiC 子系统交互的应用程序的南向入口点。
- CONFIG_DB：存储 SONiC 应用程序创建的配置状态——端口配置、接口、VLAN 等。

- STATE_DB：存储系统中配置的实体的“关键”操作状态。该状态用于解决不同 SONiC 子系统之间的依赖关系。例如，LAG 端口通道（由 teamd 子模块定义）可能指代系统中可能存在或不存在的物理端口。另一个例子是 VLAN 的定义（通过 vlanmgrd 组件），它可能引用系统中未确定存在的端口成员。本质上，这个数据库存储了解决跨模块依赖所必需的所有状态。

- ASIC_DB：存储驱动 ASIC 配置和操作所需的状态——这里的状态以 ASIC 友好的格式保存，以简化 syncd（请参阅下面的详细信息）和 ASIC SDK 之间的交互。

- COUNTERS_DB：存储与系统中每个端口关联的计数器/统计信息。此状态可用于满足 CLI 本地请求，或为远程使用提供遥测通道。

3.2.3 dhcp-relay container

DHCP 代理允许将来自没有 DHCP 服务器的子网的 DHCP 请求中继到其他子网上的一个或多个 DHCP 服务器。

3.2.4 iccpd container

此容器是一个装载着 mc-lag 功能的容器。MC-LAG 的关键是组成 MC 的两个 Chassis 之间的信息同步，SONiC 支持多种两个 Chassis 之间的互联方式，从两个 Chassis 部署在相同节点，有或者无互联专用链路，到两个 Chassis 部署在不同节点，通过各种隧道技术互联等。

3.2.5 lldp container

顾名思义，此容器承载 LLDP 功能。这些是在此容器中运行的相关进程：

- lldpd：具有 LLDP 功能的实际 LLDP 守护进程。这是与外部对等点建立 LLDP 连接以通告/接收系统功能的过程。

- lldp_syncd：负责将 LLDP 的发现状态上传到中心化系统的消息基础设施（Redis-engine）的进程。通过这样做，LLDP 状态将被传递给有兴趣使用此信息的应用程序（例如 SNMP）。

- lldpmgrd：进程为 LLDP 守护进程提供增量配置功能，它通过订阅 Redis-engine 中的 STATE_DB 来实现。

3.2.6 mgmt-framework container

管理框架是一个 SONiC 应用程序，它负责提供各种通用的北向接口（NBI），用于管理 SONiC 交换机上的配置和状态。该应用程序管理 NBI 的协调，以提供一种一致的方式来验证、应用和显示配置。我们基于此基础开发 RESTful APIs。

3.2.7 pmon container

负责运行多个守护进程，用于定期记录来自硬件组件的传感器读数，并支持告警功能。pmon 容器还承载“xcvrd”进程以及相应的光模块相关的状态。

3.2.8 radv container

此容器支持 RADVD（The Router Advertisement Daemon）功能，该进程是给系统管理员用于实现在 IPv6 下对主机进行无状态自动组态地址。当主机组态其网络接口时，会向网络多播一个路由请求来发现可用的路由，radvd 会回应一个路由广播（router advertisement，RA）的讯息。此外，radvd 会定期在网络多播 RA 包中给出连接其的链接来更新主机信息。RA 包包含了该连结的 route prefix、MTU、预设响应的路由器地址等信息。

3.2.9 sflow

sFlow 是一种基于标准的采样技术，满足交换机和路由器网络流量监控的关键要求。sFlow 使用两种类型的采样：

- 交换或路由数据包流基于统计数据包采样，以提供对网络使用情况和活动路由的可见性。
- 接口计数器的基于时间的采样。

sFlow 监控系统包括：

- sFlow 代理驻留在网络设备中，收集网络流量和端口计数器，并将流样本和接口计数器组合成 sFlow 数据报，并通过 UDP 套接字定期将它们转发到 sFlow 收集器。数据报包括但不限于包头、入口和出口接口、采样参数和接口计数器的信息。单个 sFlow 数据报可能包含来自许多流的样本。
- 接收和分析 sFlow 数据的 sFlow 收集器。

sFlow 是一种行业标准、低成本和可扩展的技术，它使单个分析仪能够提供网络范围的视图。

3.2.10 snmp container

承载 SNMP 功能。这个容器中有两个相关的进程：

- snmpd：实际的 SNMP 服务器，负责处理来自外部网络元素的传入 SNMP 节点的访问。
- snmp-agent（sonic_ax_impl）：这是 SONiC 对 AgentX snmp 子代理的实现。该子代理向主代理（snmpd）提供从集中式 Redis 引擎中的 SONiC 数据库收集的信息。

3.2.11 swss container

交换机状态服务（swss）容器由一组工具组成，允许在所有 SONiC 模块之间进行有效通信。如果数据库容器擅长提供存储能力，swss 主要侧重于提供机制来促进所有不同方之间的通信和仲裁。

swss 还托管负责与 SONiC 应用层进行北向交互的进程。如前所述，例外是 fpmsyncd、teamsyncd 和 lldp_syncd 进程，它们分别在 bgp、teamd 和 lldp 容器的上下文中运行。无论这些进程在何种环境下运行（在 swss 容器内部或外部），它们都有相同的目标：提供允许 SONiC 应用程序和 SONiC 的集中消息基础设施（Redis-engine）之间连接的方法。这些守护进程通常由所使用的命名约定来标识：*syncd。

- portsyncd：侦听与端口相关的 netlink 事件。在启动过程中，portsyncd 通过解析系统的硬件配置文件来获取物理端口信息。在所有这些情况下，portsyncd 最终将所有收集的状态推送到 APPL_DB。端口速度、信道和 mtu 等属性通过此通道传输。Portsyncd 还将状态注入 STATE_DB。
- intfsyncd：监听接口相关的 netlink 事件并将收集到的状态推送到 APPL_DB。与接口关联的新的/更改的 IP 地址等属性由该进程处理。
- neighsyncd：侦听由新发现的邻居作为 ARP 处理的结果触发的与邻居相关的 netlink 事件。诸如 mac 地址和邻居地址族之类的属性由该守护程序处理。该状态最终将用于构建数据平面中用于 L2 重写目的的邻接表。再一次，所有收集的状态最终都被传输到 APPL_DB。
- teamsyncd：之前讨论过——在 teamd 容器中运行。与前面的情况一样，获得的状态被推送到 APPL_DB。
- fpmsyncd：之前讨论过——在 bgp 容器中运行。同样，收集的状态被注入到 APPL_DB 中。
- lldp_syncd：之前也讨论过——在 lldp Docker 容器中运行。

上述流程显然充当状态生产者，因为它们将信息注入由 Redis 引擎表示的发布者-订阅者管道。但显然，必须有另一组进程充当愿意消费和重新分配所有这些传入状态的订阅者。以下守护进程正是这种情况：

- orchagent：swss 子系统中最关键的组件。Orchagent 包含提取*syncd 守护进程注入的所有相关状态的逻辑，相应地处理和发送这些信息，最后将其推送到其南向接口。这个南向接口又是 Redis 引擎（ASIC_DB）中的另一个数据库，所以我们可以看到，Orchagent 既作为消费者（例如来自 APPL_DB 的状态），也作为生产者（状态是推入

ASIC_DB）。

- intfMgrd：响应来自 APPL_DB、CONFIG_DB 和 STATE_DB 的状态，以配置 Linux 内核中的接口。此步骤仅在被监视的任何数据库中不存在冲突或不存在不一致状态时才完成。有关这种不希望的行为的示例，请参阅上述数据库容器部分。
- vlanMgrd：响应来自 APPL_DB、CONFIG_DB 和 STATE_DB 的状态，以在 Linux 内核中配置 VLAN 接口。与 IntfMgrd 的情况一样，只有在满足依赖状态/条件时才会尝试此步骤。

3.2.12 syncd container

此容器目标是提供一种机制，允许交换机的网络状态与交换机的实际硬件 ASIC 同步。这包括初始化、配置和收集交换机的 ASIC 当前状态。

这些是同步容器中存在的主要逻辑组件：

- syncd：负责执行上述同步逻辑的进程。在编译时，syncd 与硬件供应商提供的 ASIC SDK 库链接，并通过调用为此效果提供的接口将状态注入 ASIC。Syncd 订阅 ASIC_DB 以接收来自 SWSS 参与者的状态，同时注册为发布者以推送来自硬件的状态。
- SAI API：交换机抽象接口（SAI）定义了 API，以提供一种独立于供应商的方式来以统一的方式控制转发元素，例如交换 ASIC、NPU 或软件交换机。
- ASIC SDK：硬件供应商应提供驱动其 ASIC 所需的 SDK 的 SAI 友好实现。此实现通常以动态链接库的形式提供，该库连接到负责驱动其执行的驱动进程。

3.2.13 teamd container

在 SONiC 设备中运行链路聚合功能（LAG）。“teamd”进程是 LAG 协议基于 Linux 的开源实现。“teamsyncd”进程允许“teamd”和南向子系统之间的交互。

3.2.14 telemetry container

在数据中心网络的日常运行中，能够以结构化的格式高效、快速地获取网络设备的底层特征——运行状态或配置，将极大地方便分析网络状态，提高网络稳定性。除了 SNMP、syslog 和 CLI 等传统数据收集方法外，gRPC 是 SONiC 支持的用于遥测流的现代通信协议。

gRPC 系统数据遥测的实现主要基于 gNMI（gRPC 网络管理接口），并针对 SONiC 进行了定制。

3.3 Inspur 自研发模块

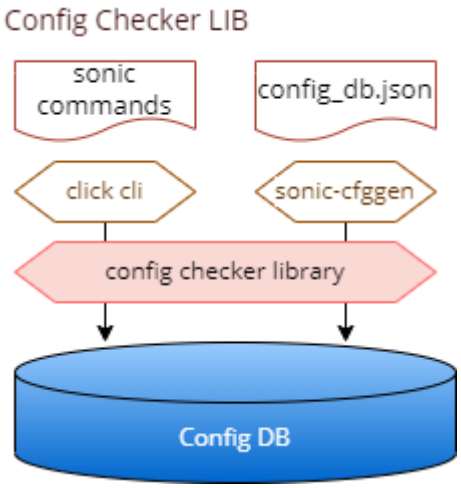
Inspur 在设计新功能时，始终遵循 SONiC 原有的流程来实现。

以下的架构图显示了相关模块之间的依赖关系以及用于该功能的数据库。详细流程请参考对应的模块 HLD 或者是对应功能白皮书说明。

3.3.1 Config checker LIB

Config checker LIB 是一个集中式的配置检查库。当配置是从 Restful、CLI 或 config_db.json 来的时候，都会经过 Config checker LIB 检查其配置是否合法。如果不合法，则会阻挡其配置下发至 Redis DB。因为是集中式检查，所以在开发效率上和品质上会提升许多。其架构图如图 3-2。

图 3-2 Config checker LIB

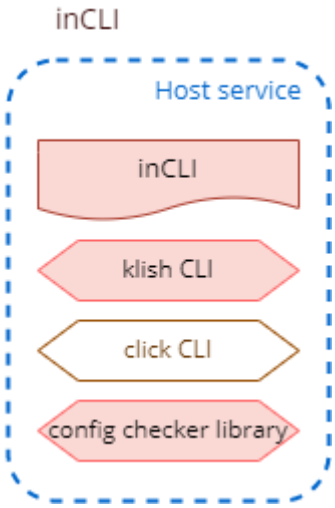


3.3.2 Klish CLI 和 inCLI command

Klish CLI engine 原本在 mgmt framework 中运行，但效能很低，为了提高效能，我们将其移出来至 host 中，成为一个独立运行的 service。

Cisco-like CLI command 是现今所有使用者都熟悉且习惯的语法，因此，在此架构中 Inspur NOS 针对所有支持的功能都实现了 inCLI commands。

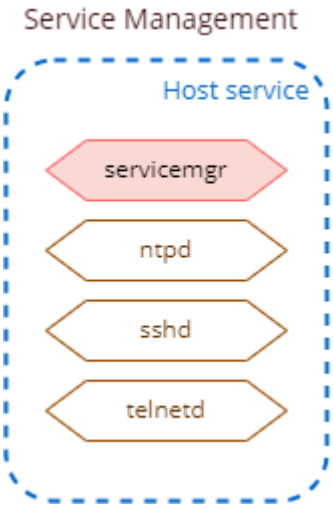
图 3-3 Klish CLI 和 inCLI command



3.3.3 Service management

这是一个远程连线管理服务，管理 NTP、Telnet、SSH 的连线服务。管理内容包含启动/关闭服务、连线数量、连线时间等等。

图 3-4 Service Management

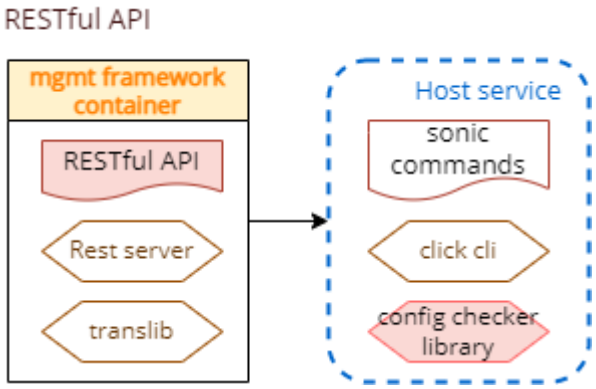


3.3.4 RESTful API

Mgmt framework 的 RESTful server 提供 Database-driven RESTful API。只有了解资料库中资料的相依性与资料库中个别资料的依赖性，才能将 Database-driven RESTful API 使用得当，意味着使用门槛很高。

Inspur NOS 在此架构之下，提供 functional-driven RESTful API。其功能是以 CLI command 为参照来实现的。如果使用者了解 CLI command 的使用，即可迅速的正确使用 functional-driven RESTful API。而如同 Cisco-like command，Inspur NOS 针对所有支持的功能也都实现了 functional-driven RESTful API。

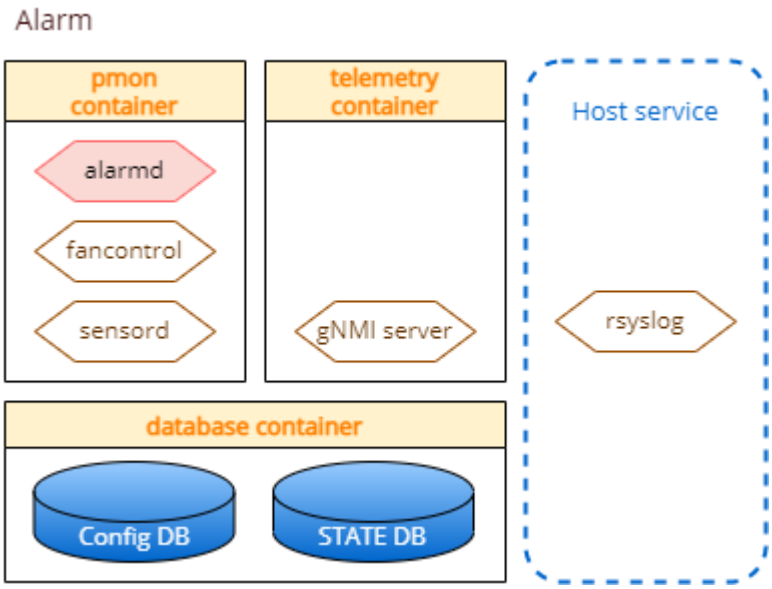
图 3-5 RESTful API



3.3.5 Alarm 告警功能

提供弹性的告警机制，包含主动式与被动式，使得系统中任何模块均可以发出告警。用户可以根据自身环境需求，配置其告警阈值。当测量数值超过阈值时，会通过 State DB 通知 telemetry container 发出告警的 gRPC packet 给使用者设定的采集器，也会将此告警记录在日志中。

图 3-6 Alarm

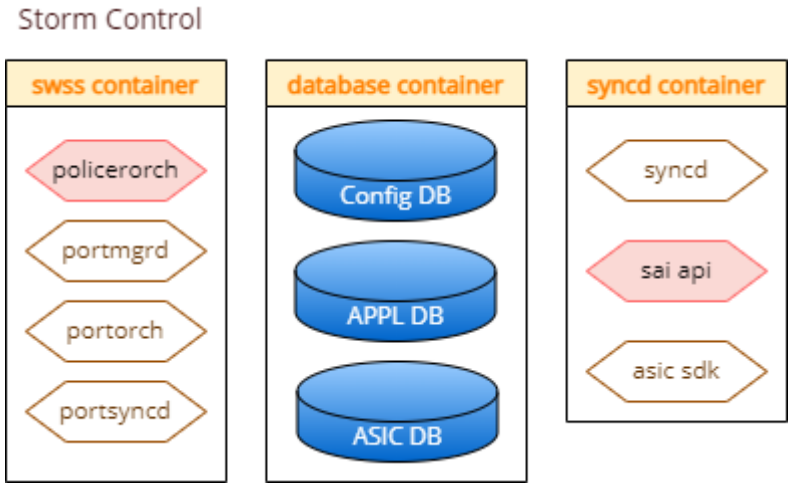


3.3.6 风暴控制功能

支持基于端口的未知单播、广播、组播三种类型风暴控制，开启端口对应报文类型的风暴控制后，相应报文速率超过阈值会发生丢包，默认基于端口入方向做风暴控制。

风暴控制需要在 ASIC 中实现才能被有效控制，所以其配置会通过 ASIC DB 下发至 syncd、SAI，然后至 ASIC 中生效。

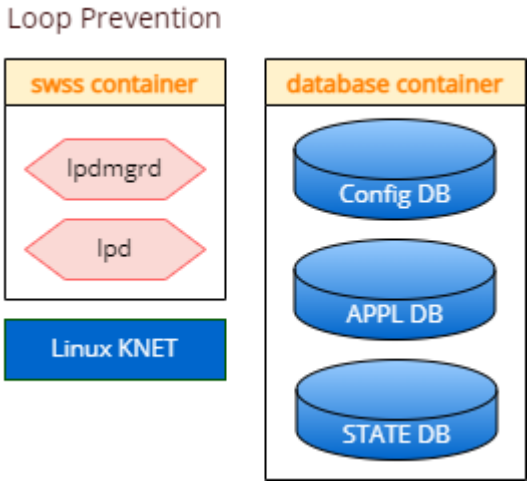
图 3-7 Storm Control



3.3.7 Loop Preventation

一般在网络上有环回产生时，将导致 packet storm，使得网络无法有效传输资料。lpd 进程的功能就是防止此网络环回的产生。lpd 藉由 Linux KNET 来发送报文，然后聆听，如果有听到此报文，则判定环回产生，将关闭此端口，以阻断环路，维持网络正常运行。

图 3-8 Loop Prevention

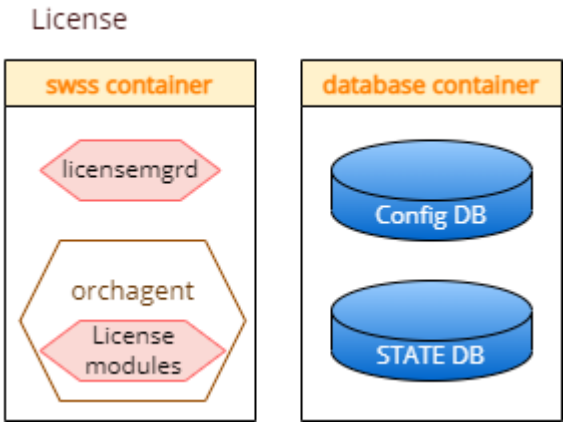


3.3.8 License

Inspur NOS 提供了一个授权机制。此机制有下列几种授权方式：试用授权、标准授权、进阶授权与功能授权。

Licensemgrd 管理来自于 config DB 的配置与用户安装的 license file。License module 则根据 license file 的定义来控制交换机功能可以使用还是禁止使用。

图 3-9 License

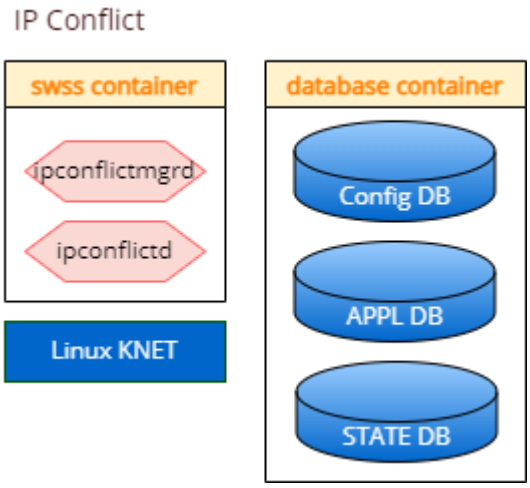


3.3.9 IP 冲突辨别

在资料中心里，每个交换机的端口下可能连接多台 server，或是 server 之中运行多个虚拟机。在此场景中，这些 server 或是虚拟机可能会因为获取或被设定成相同的 IP 地址而产生冲突。而发生冲突的 server 或虚拟机，将无法正常运行。ipconflictd 进程就是为了让资料中心管理员可以轻松得知是否有此状况发生，以迅速解决问题。

Ipconflictd 在 KNET 中聆听 ARP packet，并利用 MAC address 来判断其 IP address 是否发生冲突。

图 3-10 IP Conflict



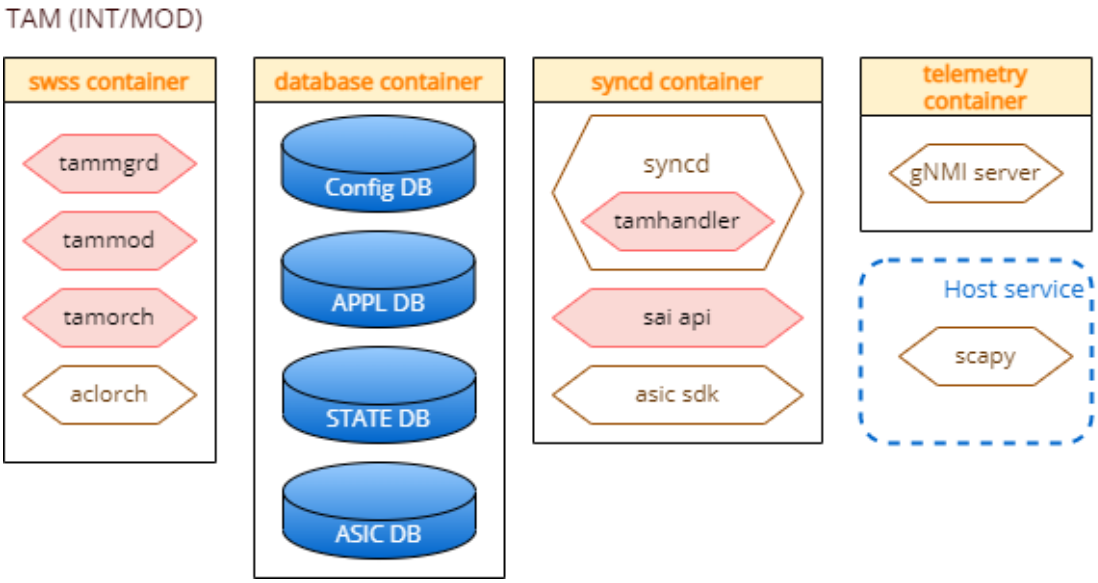
3.3.10 TAM

因为资料中心的流量越来越大，服务应用对延迟的要求越来越低，所以对网络状态的掌握度需求也越来越高。TAM、Dtel or INT 都是此类的监控功能，用来监控网络 route path、forwarding delay、packet drop、disconnect hop 等等。

Inspur NOS 提供 INT（In-band Network Telemetry）与 MOD（Monitor on Drop）的功能。利用 scapy 与 telemetry container 的功能将 meta data 封装，传送至使用者设定的

采集器，以供使用者实时了解其网络的运行情况，及时调整其网络配置或解决问题，使得其网络效能可达到最高效益。

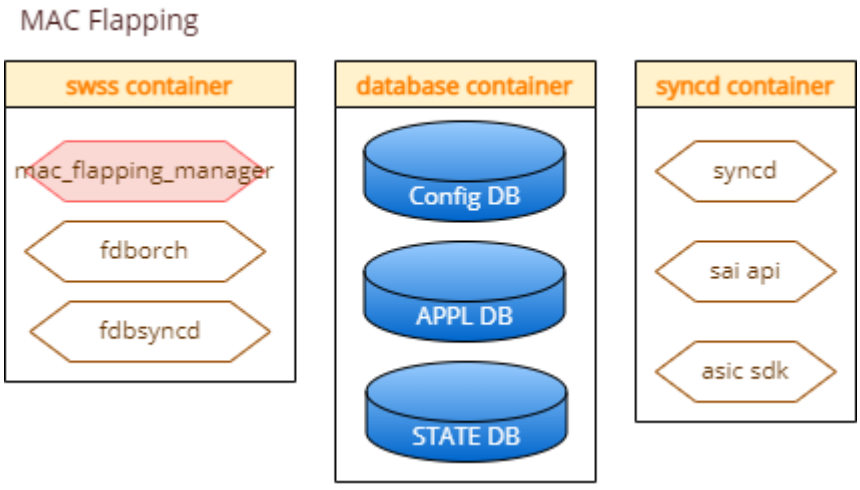
图 3-11 TAM



3.3.11 MAC Flapping

MAC 地址漂移是指设备上一个 VLAN 内有两个端口学习到同一个 MAC 地址，后学习到的 MAC 地址表项覆盖原 MAC 地址表项的现象。正常情况下，网络中不会在短时间内出现大量 MAC 地址漂移的情况。出现这种现象一般都意味着网络中存在环路，可以通过查看告警信息和漂移记录，快速定位和排除环路。InspurNOS 提供了 mac-flapping 检测功能的开启关闭功能，并且当 mac 地址飘移现象发生时，通过在接口上配置漂移检测动作为错误关闭端口，自动关闭掉漂移的端口并且记录相关日志。

图 3-12 MAC-Flapping



3.3.12 Black-Hole Route

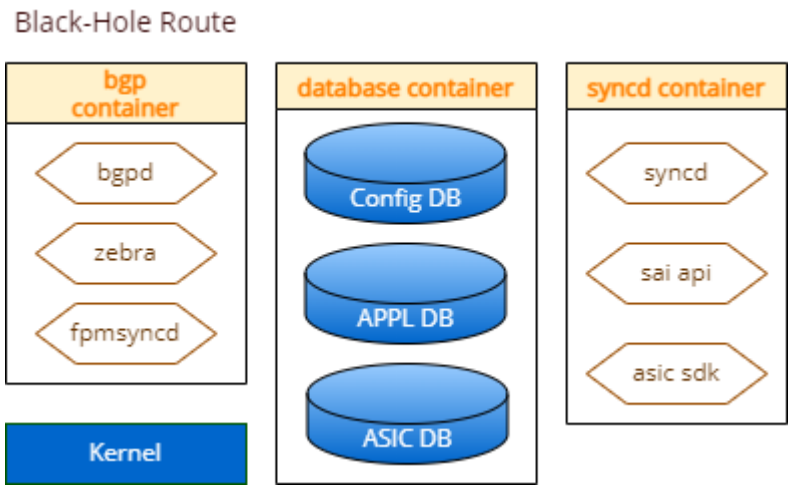
一般而言，一条路由无论是静态或者动态，都需要关联一个出口，路由的出接口可以为设备的物理口或者逻辑接口。有一种特殊的逻辑接口为 NULL0，这是一个系统保留的逻辑接口，当设备在转发某些数据包时，使用了出接口为 NULL0 的路由，那么这些报文将被直接丢弃，因此出接口为 NULL0 的路由被称为黑洞路由。

当去往某一目的地的静态路由具有黑洞路由属性时，无论配置的下一跳地址是什么，该路由的出接口均为 NULL0 接口，任何前往该目的地的 IP 报文均被丢弃，并且不通知源主机。

当配置两个普通路由可能会形成 ECMP，但是如果其中一个是黑洞路由时，黑洞路由和普通路由等两条路由都会失效。这是继承来自 FRR 中的路由管理者 zebra 的规则决定的。

在应用场景中，当普通路由无效后，可能会导致路由环路的问题，造成网络故障，对所有可能因为中断故障产生路由环路的路由都加上一条黑洞路由可以有效的避免路由环路问题。

图 3-13 Black-Hole Route



3.3.13 L2 EVPN VXLAN

EVPN（Ethernet Virtual Private Network）是下一代全业务承载的 VPN 解决方案。EVPN 统一了各种 VPN 业务的控制面，利用 BGP 扩展协议来传递二层或三层的可达性信息，实现了转发面和控制面的分离，相对于传统 VLAN，EVPN VXLAN 突破了 4094 限制。

EVPN 通常被配置在供应商边缘交换机（PE）为了实现客户在逻辑上的网络隔离，供应商边缘交换机直接连到客户端设备（CE），可以是路由器，交换机，或是主机服务器供应商边缘交换机透过 Multiprotocol BGP（MP-BGP）协议来互换路由可达信息并且封装流量转发到其他台供应商边缘交换机。

Virtual Extensible LAN protocol（VXLAN）支持网络中的更多的 VLANs，根据 IEEE 802.1Q，L2 VLAN ID 上限是 4094，VXLAN 利用 VNI（VXLAN 网络识别）和 VLAN 匹配克服了传统

上的限制，巨型的网络数据中心提供了更多的逻辑上网络隔离。VXLAN 引入 overlay 方案，将 L2 网络地址空间从 4K 扩展到 1600 万 VXLAN 隧道协议，将 L2 以太网帧封装在 L3 UDP 数据包中。这种 VXLAN 封装能够创建可以跨越物理第 3 层网络的虚拟第 2 层子网或分段 VNI 的引进就像是 IEEE 802.1Q VLAN ID，承载相同 VNI 的设备可以直接相互通信，而承载不同 VNI 的设备需要路由器才能相互通信。

L2 EVPN VXLAN 为实现 Overlay 网络拓扑虚拟 L2 封包相关行为。

图 3-14 L2 VxLAN Tunnel Creation

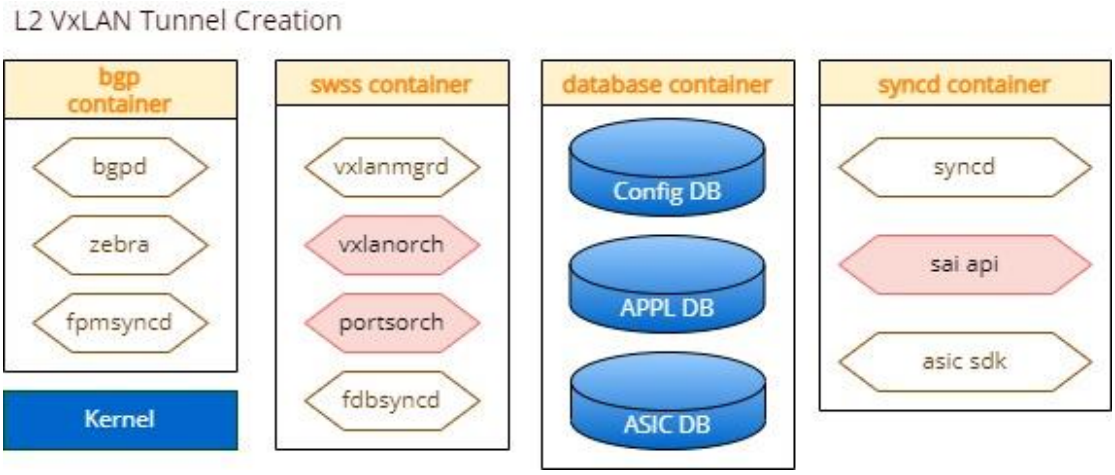


图 3-15 Local MAC Learn

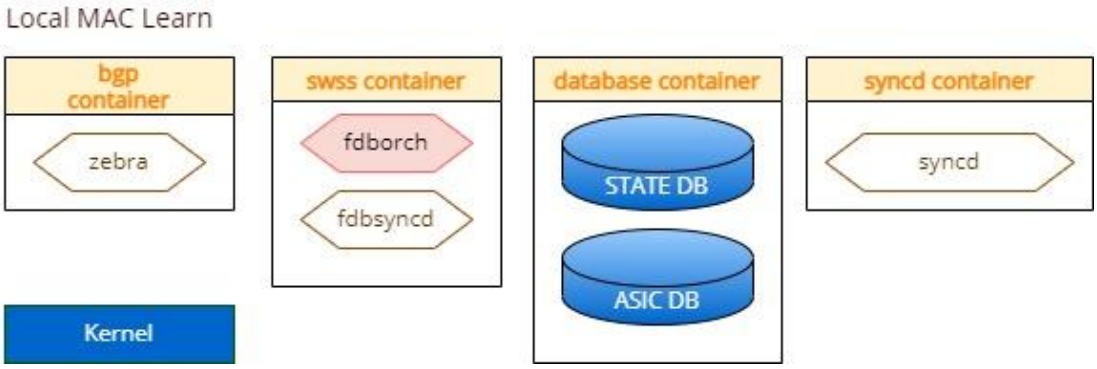


图 3-16 Remote MAC Learn

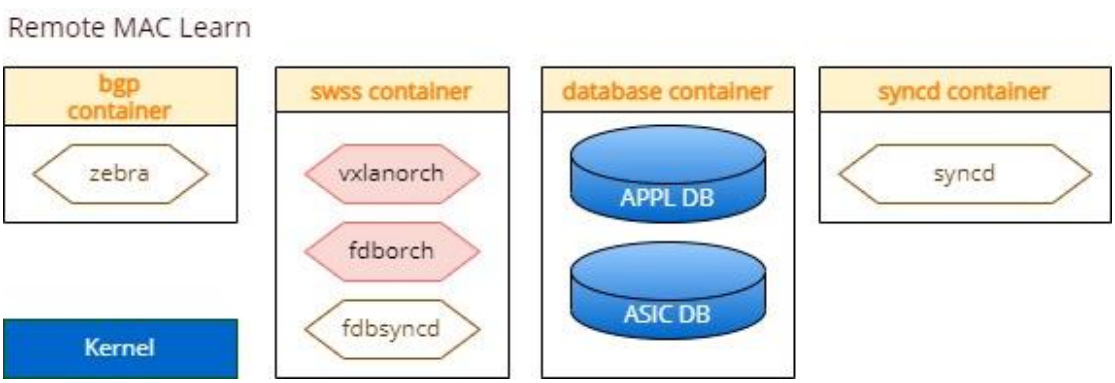
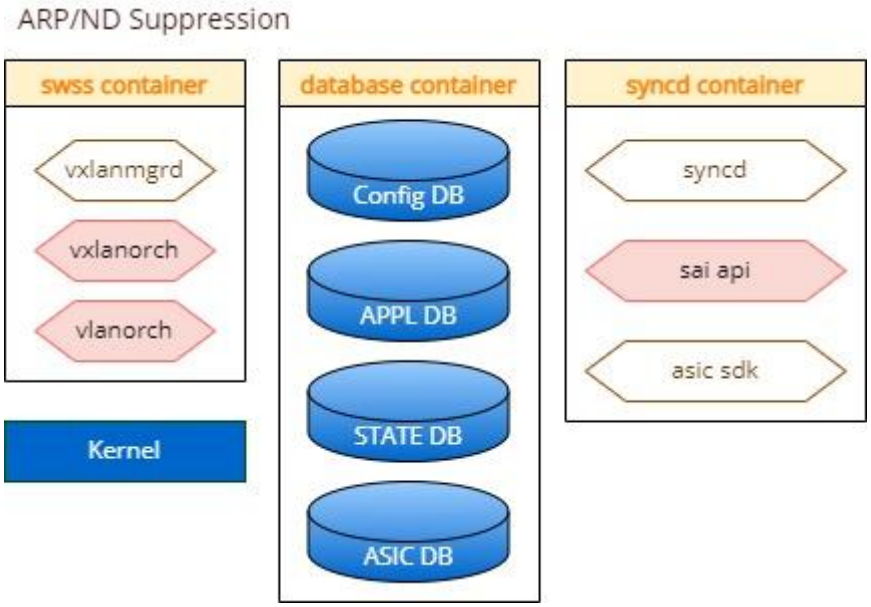


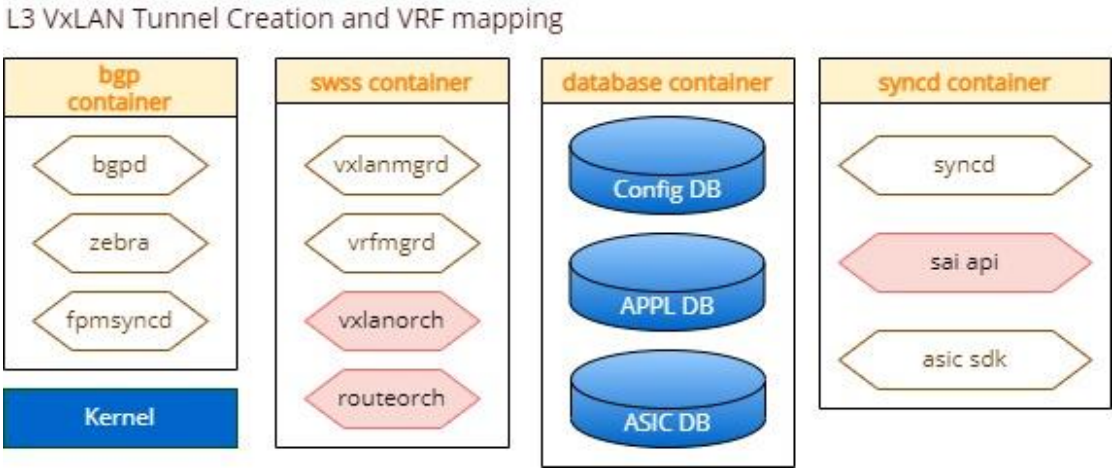
图 3-17 ARP/ND Suppression



3.3.14 L3 EVPN VXLAN

承上述 EVPN VxLAN 介绍，L3 EVPN VXLAN 为实现 Overlay 网络拓扑虚拟 L3 跨网段封包相关行为。

图 3-18 L3 VxLAN Tunnel Creation and VRF mapping



3.3.15 Policy-Base Route

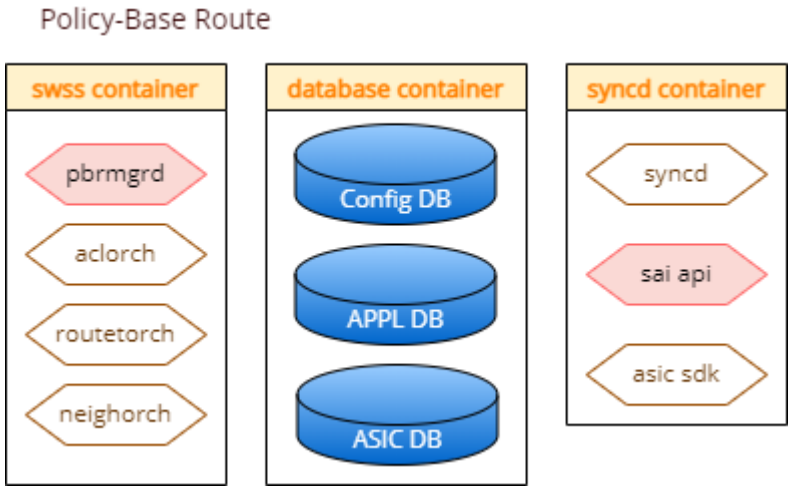
策略路由（英语：Policy-based Route，缩写为 PBR），一种决定路由的方式，由网络管理者决定路由规则，再根据这些规则来制定路由。

当一个路由器接收到封包时，通常会被转送到封包指定的目的地址。但在某些状况下，需要根据其他规则来决定封包要转送到何处。举例来说，网络管理员可以根据封包的来源地址来转送这些封包。

策略路由可以根据封包的大小，封包内指定的通讯协议，或是其他封包表头及封包内容的信息，来决定路由转送的方式。当有数个私有网络相互链接时，策略路由对于网络管理员来说是相当有用的。

Inspur NOS 的策略路由让用户可以根据封包的源地址、目的地址、源端口、目的端口、DSCP 等字段来制定封包转传的下一跳。用户配发完策略路由后会透过 pbrmgrd 将其转成 ACL 规则并写入 APPL DB，然后 aclorch 会通过 ASIC DB 下发至 syncd、SAI 后生效，硬件就可以根据 ACL 规则直接转发封包。

图 3-19 Policy-Base Route

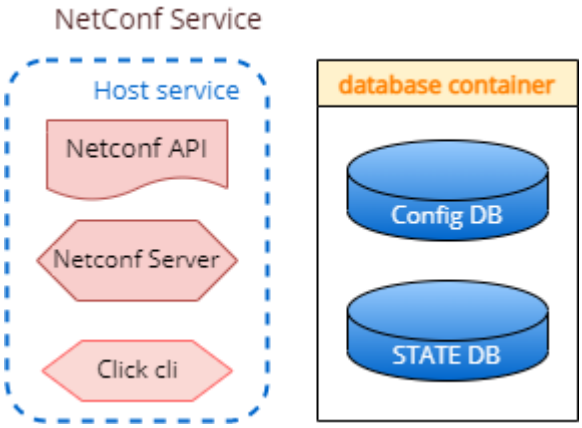


3.3.16 NetConf

在早期的数据中心内，由于网络环境相对简单，网管人员通常使用 CLI（Command Line Interface）来配置网络设备。随着数据中心日渐复杂，网络设备越来越多，网络自动化需求的崛起，CLI 的配置方式显然已成为了一种瓶颈。

Inspur NOS 提供 NETCONF（Network Configuration Protocol，网络配置协议）功能，提供用户一种安全、高效且可程序化的配置设备方式

图 3-20 Netconf Service



3.3.17 sFlow

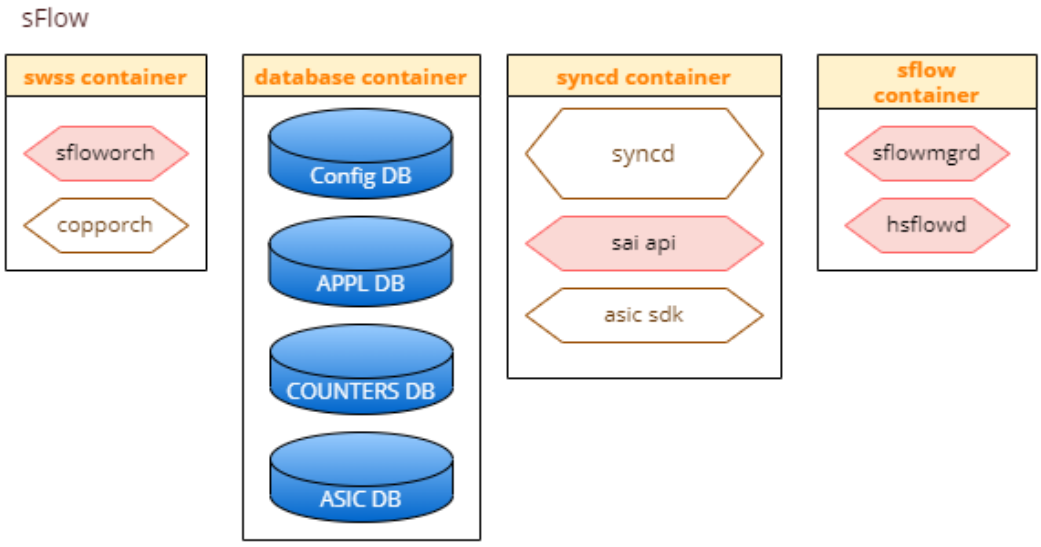
网络管理一直是从小到大各种规模的公司长期追求的目标。努力理解所有可能的传输流量、带宽需求、性能含义、安全威胁和计费分配仅仅是当今网络管理人员所面临的挑战中的很小一部分。随着网络在规模、速度和容量方面的发展，采用基于 RMON 或 NetFlow 计数器和统计的传统工具进行监视和管理变得越来越困难。

sFlow 是一种基于标准的采样技术和流量监控技术，用于收集和分析流量统计信息，可满足对交换机和路由器的网络流量监控的关键要求。sFlow 使用两种类型的采样：

1.交换或路由数据包流的基于统计数据包的采样，以提供对网络使用情况和活动路由的可见性

2.接口计数器的基于时间的采样

图 3-21 sflow

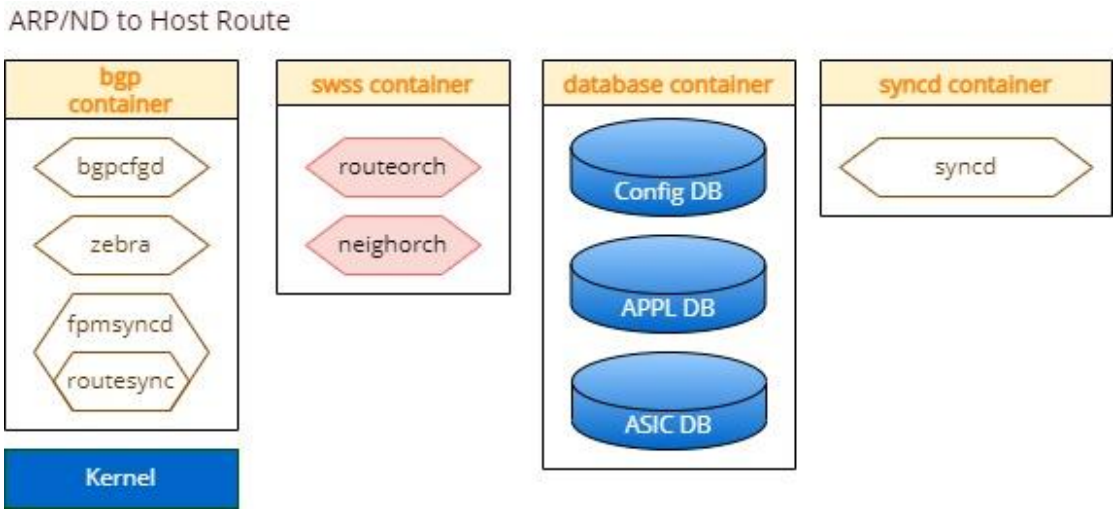


3.3.18 ARP/ND to Host Route

开启 ARP 转直连路由功能，当 Switch 学习到 ARP 信息以后，会生成 32 位掩码的直连主机路由 x.x.x.x/32。在 Switch 的 BGP 路由协议中配置引入该直连主机路由（例如通过 network 命令），将此路由发布给其他 Switch。

ND 转直连路由的原理跟 ARP 转直连路由类似，会根据学习到的 IPv6 ND 表项生成 128 位掩码的直连主机路由。

图 3-22 ARP/ND to Host Route



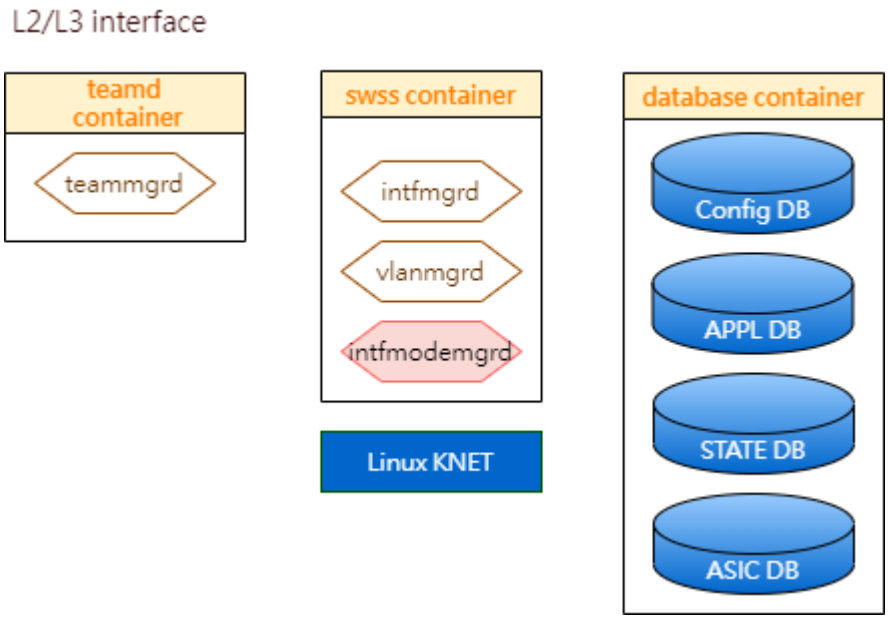
3.3.19 L2/L3 interface definition

L2/L3 interface 接口技术是一种 Cisco-like CLI Command 的解决方案，以用来区分 Ethernet 和 Port-channel 接口二层和三层的状态，并且定义非二层非三层接口状态，技术上可以明确的区分出 Ethernet 和 Port-channel 接口的状态，并且在各个状态之间转换。

intfmgrd、vlanmgrd，以及 teammgrd 管理 CONFIG DB Ethernet 和 Port-channel 的配置，写入 APPL DB，最终写入 ASIC DB 中生效。

intfmodemgrd 根据 CONFIG DB 配置，将接口状态写入 STATE DB。

图 3-23 L2/L3 Interface

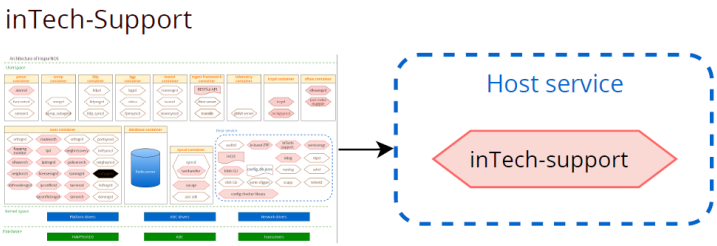


3.3.20 inTech-Support

inTech-Support 提供一键收集方法，能方便且快速的收集交换机上的关键信息，用于数据分析与信息汇集，并可指定信息收集后自动存于本机端 USB 装置或传输至远程装置。

inTech-support 另外提供一键收集或汇入交换机上的配置文件。

图 3-24 inTech-Support

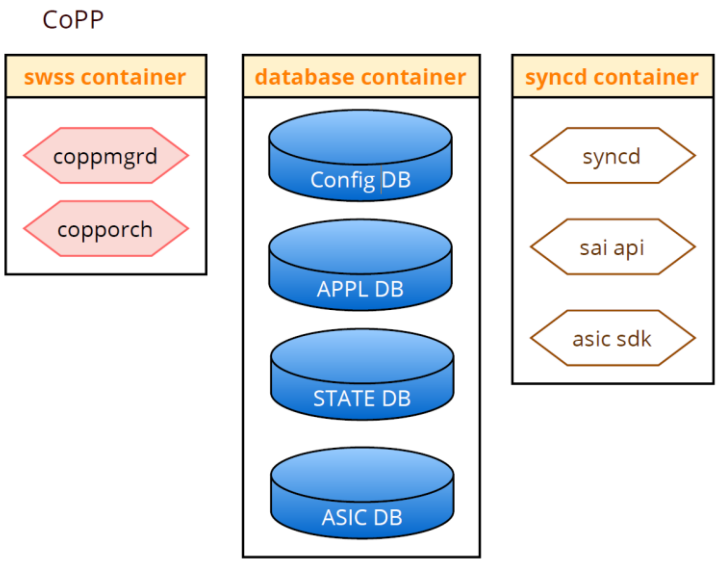


3.3.21 CoPP Dynamic Configuration

针对 Control Plane 来进行网络联机封包的数据分析及避免 Denial of Service 所产生的一种保护机制。CoPP 特色允许使用者设定 QoS filter，以限流的方式来管理 Control Plane 封包的资料流，利用这种方式，不论在有攻击或大量资料负载时，Control Plane 仍可以协助维护封包转送及协定状态。

coppmgrd 管理来自于 Config DB 的配置，统整相关配置后写入 APPL DB。copporch 把来自于 APP DB 的信息，通过 ASIC DB 下发至 syncd、SAI，然后至 ASIC 中生效。

图 3-25 CoPP Configuration



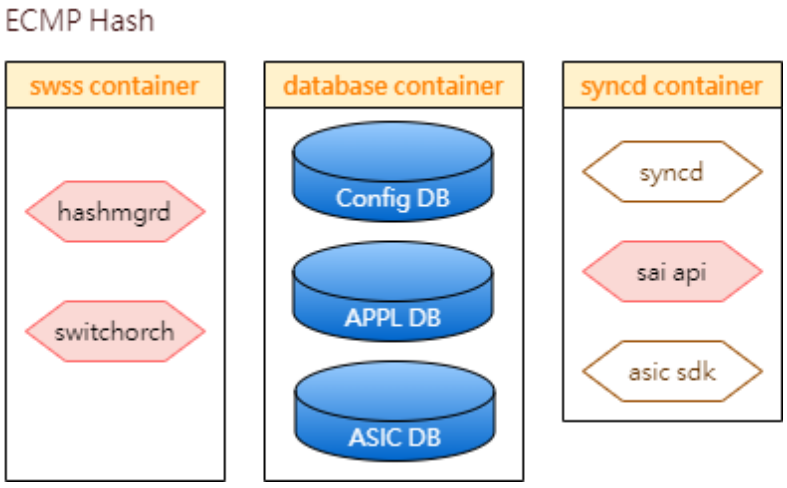
3.3.22 ECMP Hash

数据中心的网络拓扑采用的结构中，主机之间通常存在多条路径，然而为满足吞吐量的需求会提供大量的带宽资源，对于网络设备的可靠性、资源利用率日益升高，将数据流分布到不同路径上进行数据传输，避免拥塞，提高数据中心内的资源利用率。

ECMP Hash 可利用五字节以及哈希因子，来提高了网络资源利用率。

hashmgrd 透过 CONFIG DB 管理五字节、哈希因子配置，写入 APPL DB，switchorch 管理 APPL DB 最终写入 ASIC DB 中生效。

图 3-26 ECMP Hash



3.3.23 Collector

基于统一前端 Server（Envoy）的反向代理和负载均衡，提供统一的前端 http 服务，包括：

- 可视化展示 UI 服务

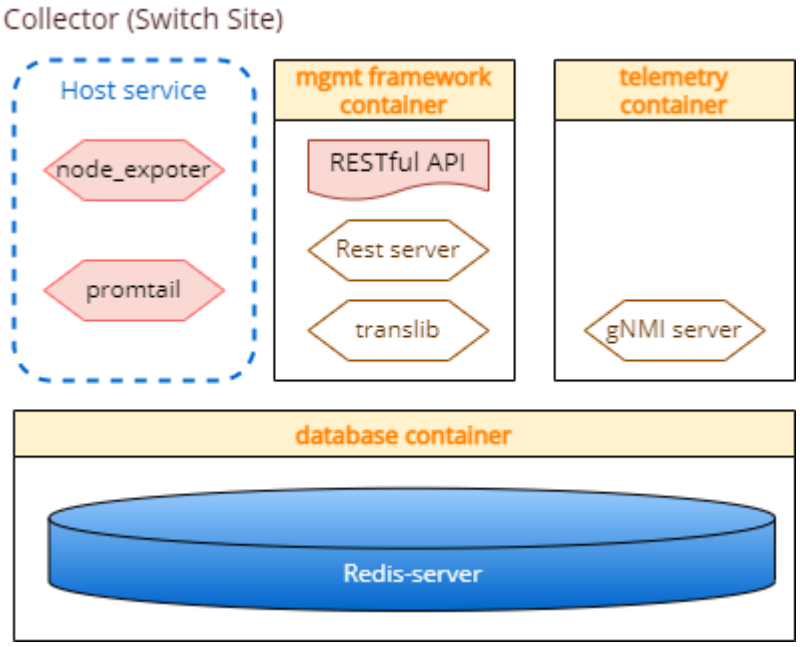
- 可视化配置服务
- 纳管交换机节点的 NBI 管理配置 RESTful API 服务
- 时标数据采集服务的 NBI 查询 RESTful API 服务
- 日志采集服务的 NBI 查询 RESTful API 服务

除了以上北向服务接口外，还有以下可选的南向服务接口：

- 时标数据采集服务的 Promethus 的 pushgateway 南向服务接口日志采集服务的南向采集服务接口
- APPs 数据采集服务的南向通信接口

从安全上考虑，南向服务接口可选是否通过 Envoy 反向代理，无论实现选择上，是否通过 Envoy，南向连接都需要实现区别于北向服务的安全访问权限控制。

图 3-27 Collector

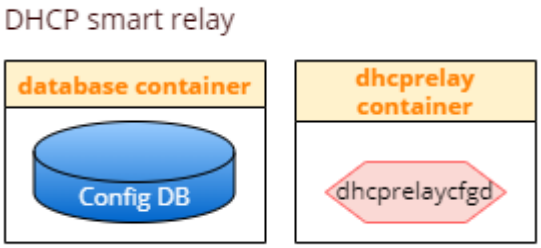


3.3.24 DHCP Smart Relay

当没有来自 DHCP 服务器的 DHCP-OFFER 消息时，允许 DHCP 中继代理将网关地址（DHCP 数据包的 giaddr 字段）切换到辅助地址。

如果配置了 ip dhcp smart-relay 命令，当 DHCP 服务器没有 DHCP-OFFER 消息时，中继代理计算客户端重试向 DHCP 服务器发送请求的次数。重试 3 次后，中继代理将网关地址替换为辅助地址。如果 DHCP 服务器在重试 3 次后仍然没有响应，则使用下一个辅助地址作为网关地址。

图 3-28 DHCP smart relay



3.3.25 Dynamic Load Balance

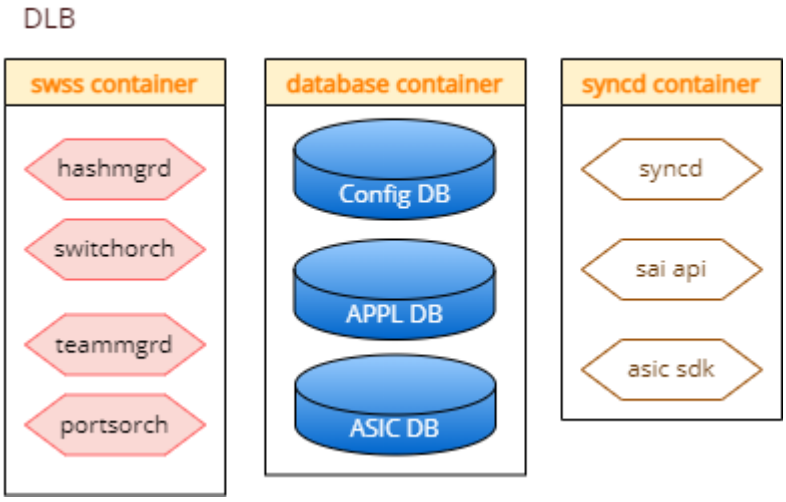
动态负载平衡（Dynamic Load Balance，DLB）是用来解决传统静态哈希的多路径冲突及防止突发的大象流占据老鼠流的情况发生。将一段突发数据流称为 Flowlet，Flowlet 的报文间隔小于 Flowlet gap time，Flowlet 仍会走相同路径确保报文顺序不会混乱，反之则会执行动态负载平衡，选择新的路径进行数据传输。

DLB 分为 ECMP mode 和 Trunk mode。

设定 ECMP mode 时，由 hashmgrd 根据 CONFIG_DB 往 APPL_DB 设定 DLB 相关参数，switchorch 根据 APPL_DB 变化写入 ASIC_DB。

设定 Trunk mode 时，由 teammgrd 根据 CONFIG_DB 往 APPL_DB 设定 DLB 相关参数，portsorch 根据 APPL_DB 变化写入 ASIC_DB。

图 3-29 DLB

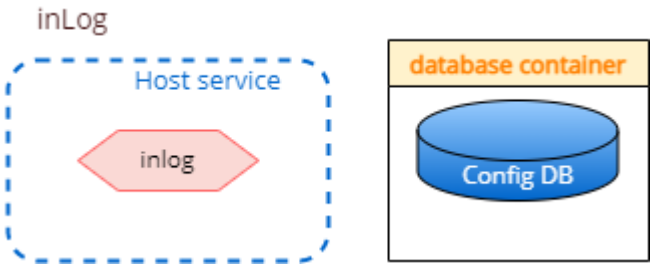


3.3.26 inLog

传统的 rsyslog 要修改内容必须去修正/etc/rsyslog.conf，并且知道所使用的 module 是否有加载以及该如何使用此 module 的设定。

Inspur NOS 以 rsyslog 的功能为基底，将之整合成 inlog 指令，所以我们可以透过 inlog 指令直接设定，inlog 会链接 config_db 并搜集 config_db 的信息，并透过 shell 去产生新的 rsyslog config 并重启 rsyslog service，使用者不需要手动去修改/etc/rsyslog.conf，并且所有指令都有 help message，可以知道指令的作用为何，以及需要加入那些参数。

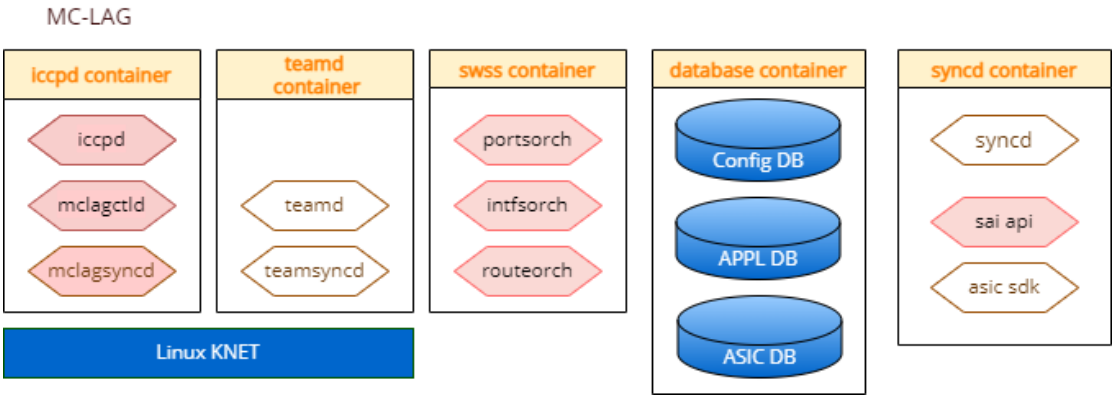
图 3-30 inLog



3.3.27 MC-LAG

MC-LAG 在完成配置以后，会通过比较 ipv4 地址大小确定主备关系，地址大的为备设备，小的为主设备。主设备会主动使用 TCP 与备设备建立邻居关系。在两台设备建立起邻居关系以后，备设备会把本设备上所有 MC-LAG 成员端口的 MAC 地址修改为与主设备的 system MAC 相同，这样在进行 LACP 协商时，主、备设备发送的 LACP PDU 中包含相同的 system id，使对端设备感觉是在与同一台设备进行协商，从而达到跨设备进行链路聚合的目的。邻居关系建好以后，设备之间会进行 FDB MAC、ARP 等信息的同步。通过控制 FDB、ARP 表项下发的端口，可以控制单播报文的转发通道，即在正常情况下，peerlink 只作为备份链路不转发流量，而当出现故障时，流量可以从 peerlink 转发，从而实现流量不会因为故障而中断。对于 BUM 报文，MC-LAG 设计了单向隔离机制，以保证同一台设备不会收到 BUM 报文的多份拷贝。

图 3-31 MC-LAG

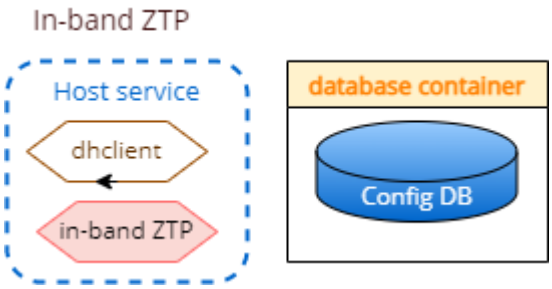


3.3.28 In-band ZTP

用户可以使用零接触配置（ZTP）服务使用通用配置模板来配置一组交换机。从出厂默认

状态启动的交换机应该能够与远程配置服务器通信并下载相关配置文件和脚本以启动更复杂的配置步骤。ZTP 服务接受 JSON 格式的用户输入。除了原生 SONiC 小区提供的带外网络 Out-band 的 ZTP 服务，浪潮也增加了带内网络 In-band 的 ZTP 服务，具备弹性且满足大量部署的需求。在其 In-band ZTP 当中，浪潮也提供自适应速率的方法，主动调适与提供配置 Server 之间的速率。

图 3-32 ZTP



4 主要特性

图 4-1 Inspur NOS 协议栈

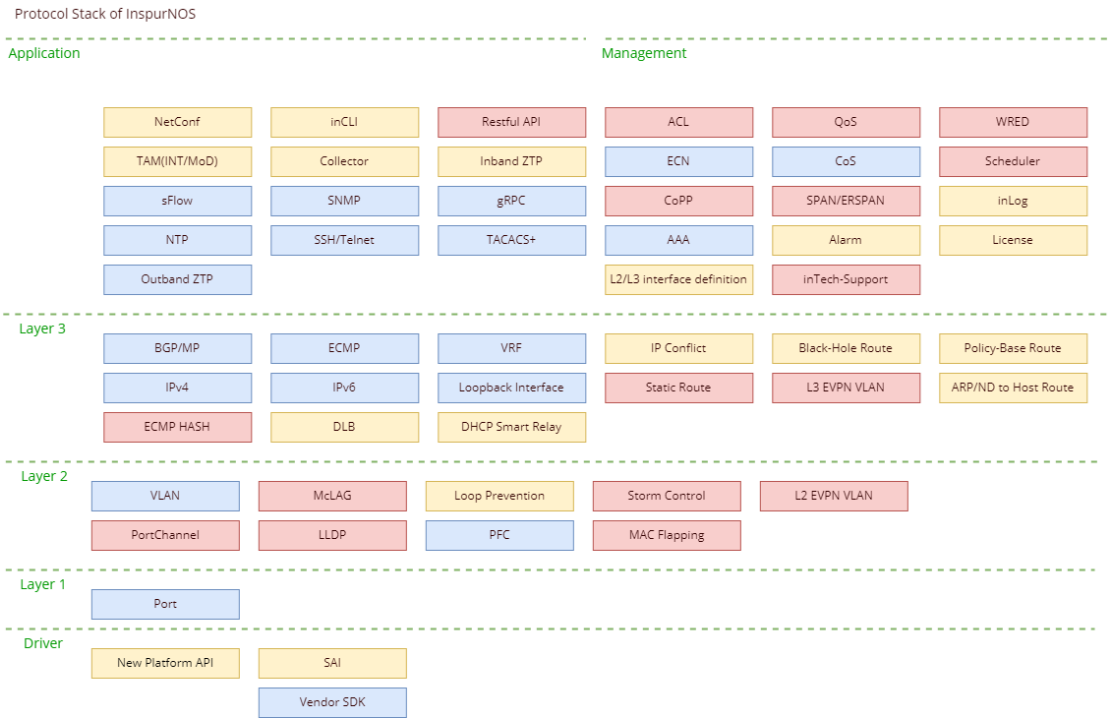


图 4-1 根据阶层，列出 Inspur NOS 所支持的协议。

- 蓝色的部份，是原生 SONiC 所支持且 InspurNOS 修改较少的模块协议。
- 红色的部分，是 InspurNOS 增强 SONiC 的模块协议。
- 黄色的部份，是 InspurNOS 自研发的模块协议。