DaoliNet

A Simple and Smart Networking Technology for Docker Applications

DaoliNet

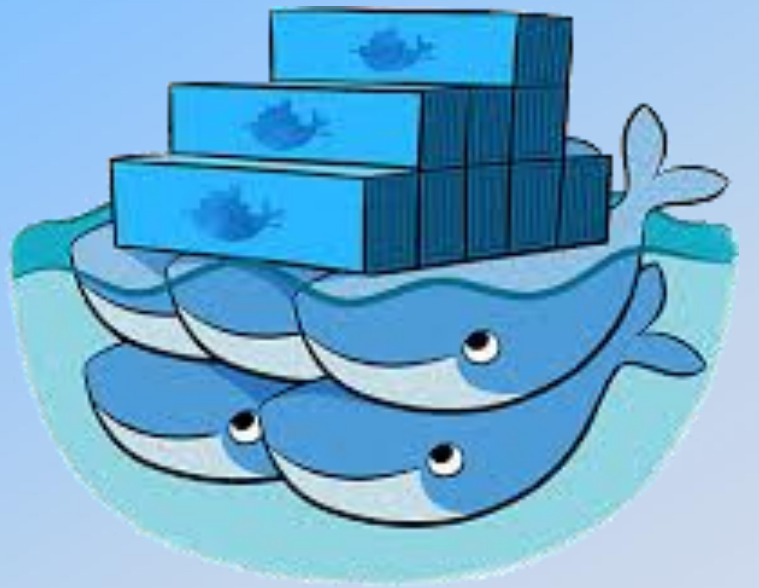
An Open Source Project

www.daolinet.org

May, 2016

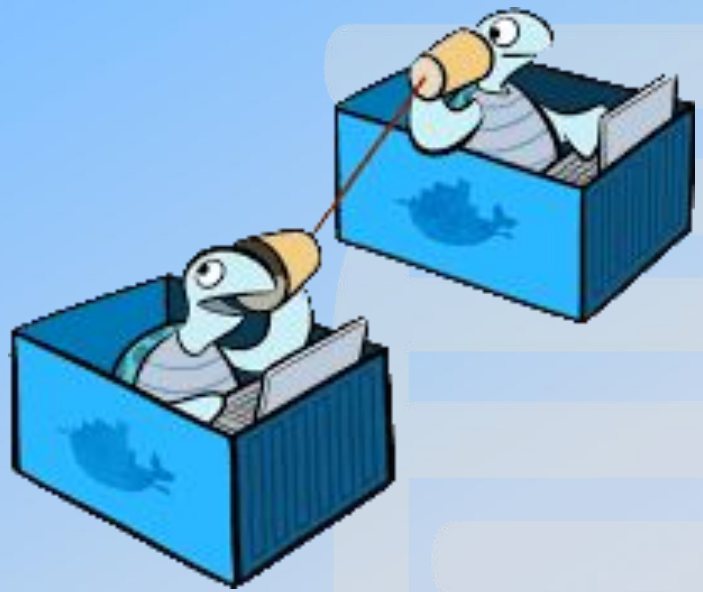
Docker is Awesome!

- A Linux Container Engine
- Build, Ship and Run Any App Anywhere
- Fast Delivery, Easy Portable, High Density of Workloads



Docker Networking Question

- Every Docker host is created independently, therefore containers in different Docker hosts are not connected one another by default
- There are several Docker networking projects, Weave, Flannel, Libnetwork, Calico, etc., all consume Docker servers' resource even when containers are not in communication, e.g., Calico: servers keep running heavyweight routing algorithms; MAC-in-UDP encapsulation: MAC learning by flooding in big scale



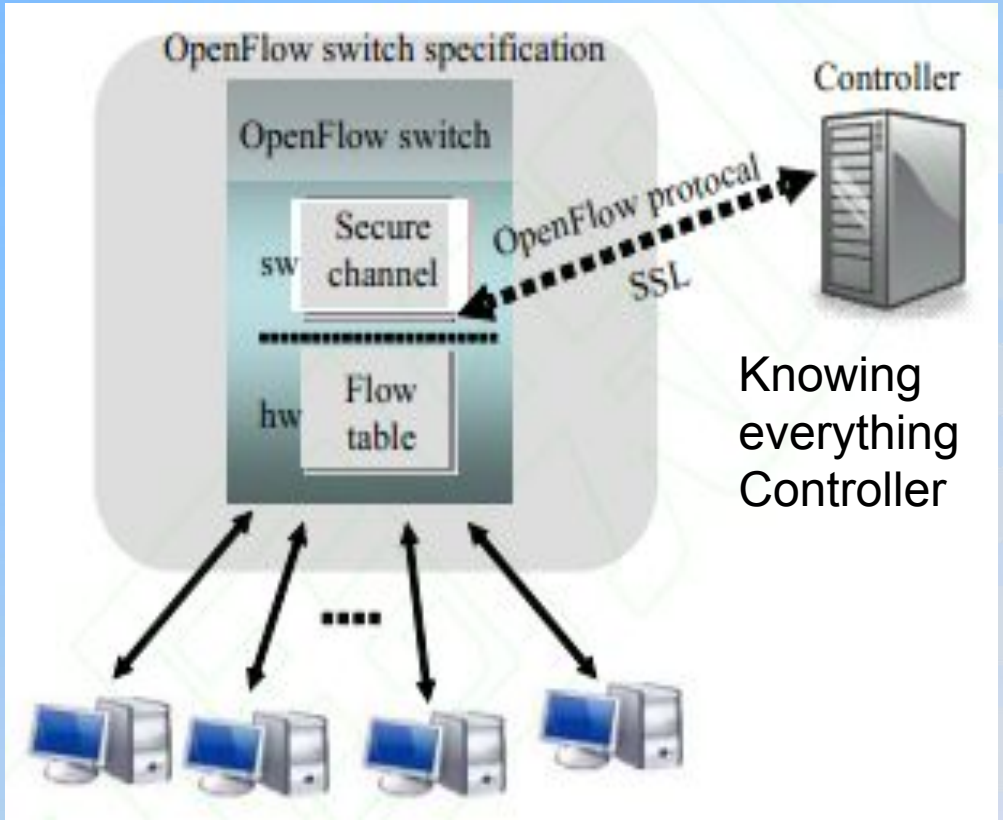
OpenFlow: Smart Network Control

OpenFlow: From Clean Slate Program at Stanford University, for Software Defined Network (SDN)

In Standardization by Open Network Foundation (ONF)

Separation of control and forwarding planes:

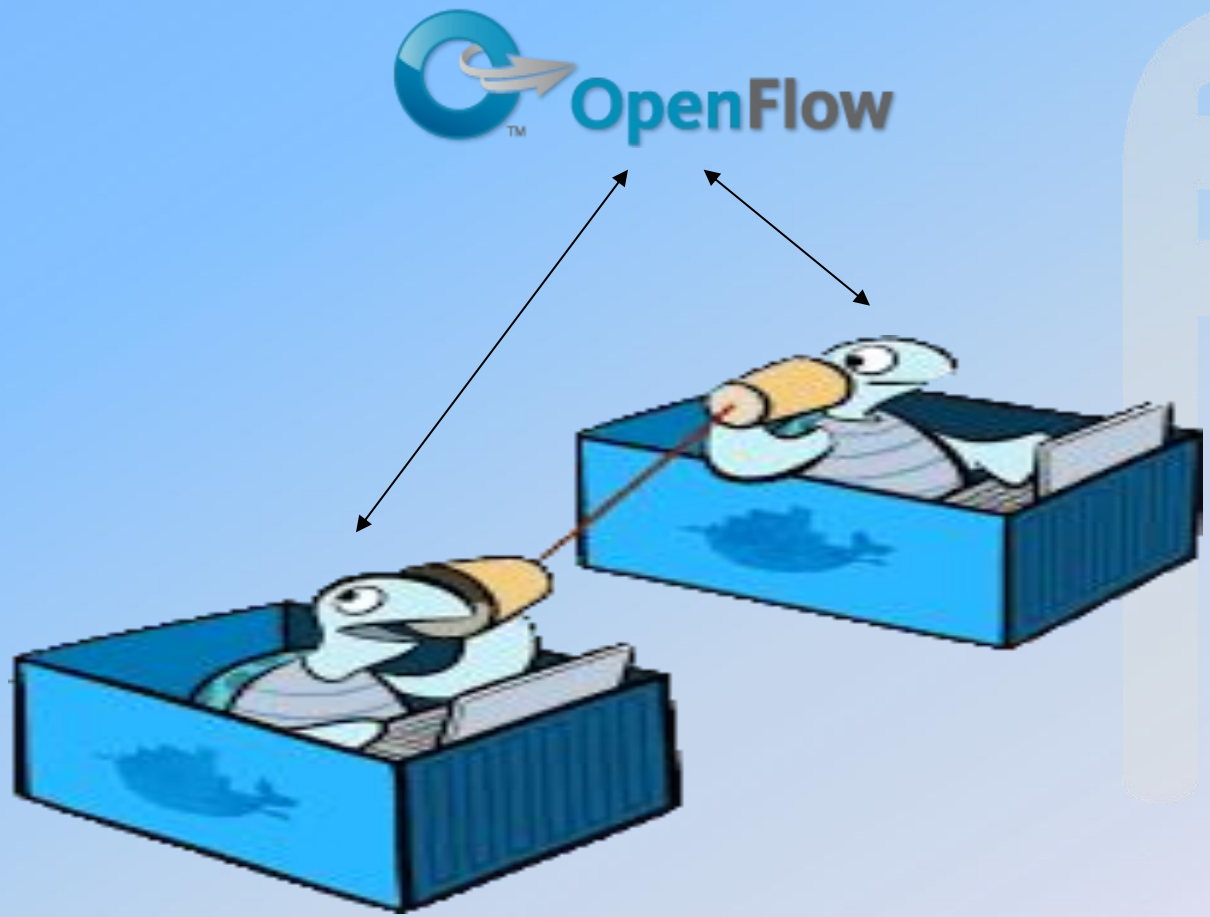
- Forwarding devices do not know each other;
- Let a knowing-all Controller tell forwarding devices what to do, in real time of communications



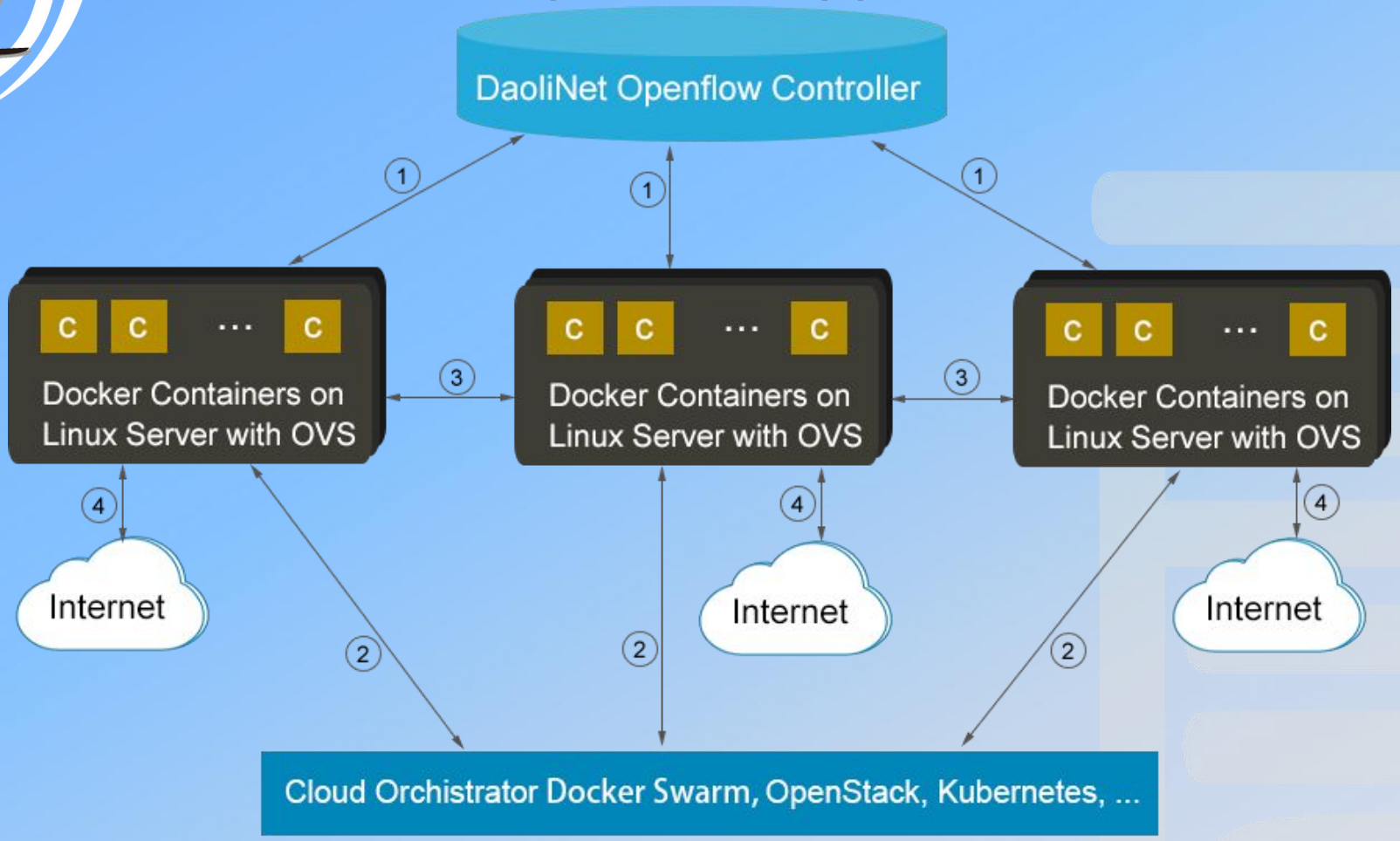
Network forwarding devices: they do not know each other

OpenFlow for Docker Networking
Keep it Simple!

Let Docker servers not know each other, leave communications open, and let a Controller tell Docker servers how to forward packets when containers start to communicate



DaoliNet: An Openflow Approach



1. Openflow Packet-in & Packet-out
2. Cloud Orchistration
3. Ethernet or VPN connected
4. Distributed Gateways & Firewalls

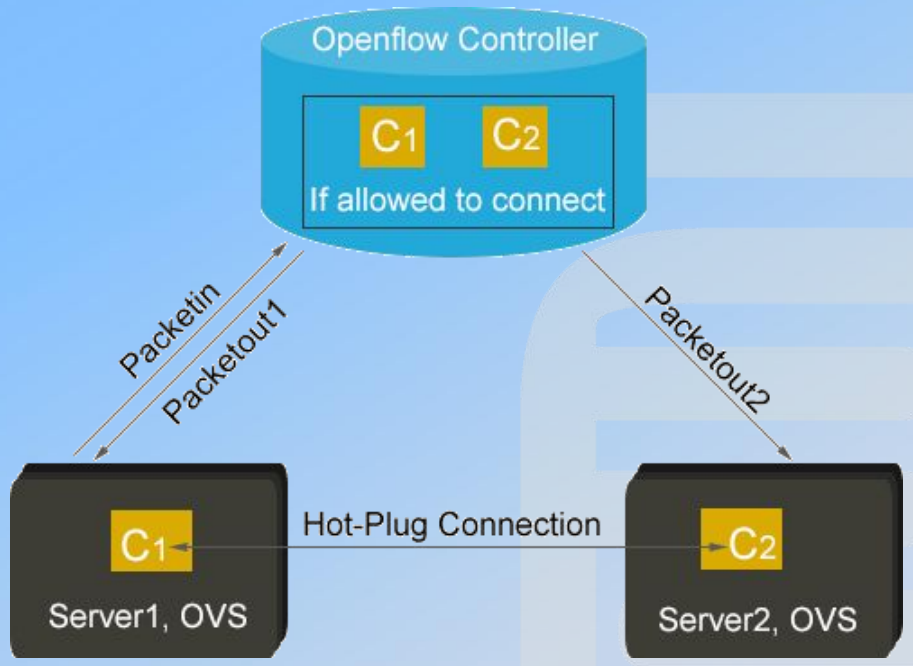
Ubiquitously distributed Open-V-Switch in Docker hosts: They will ask the Controller what to do upon containers start talk

How it Works

What Controller can see from PacketIn:

- src-ip = ID of communication initiator (C1)
- dst-ip = ID of communication responder (C2)

These IDs suffice Controller to judge legitimacy of the connection request



Controller can PacketOut two flows to OVSes in Server1 & Server2

These two flows, plus the Ethernet connection between the two servers, form a hot-plug route for the requested connection

Hot-plug connection is resource efficient, fine granular, and simple!

Overlay Network No Packet Encapsulation

When a packet travels between a container and its Docker server, src-ip, dst-ip are those of containers

When a packet travels between two Docker servers, src-mac, dst-mac are those of the servers

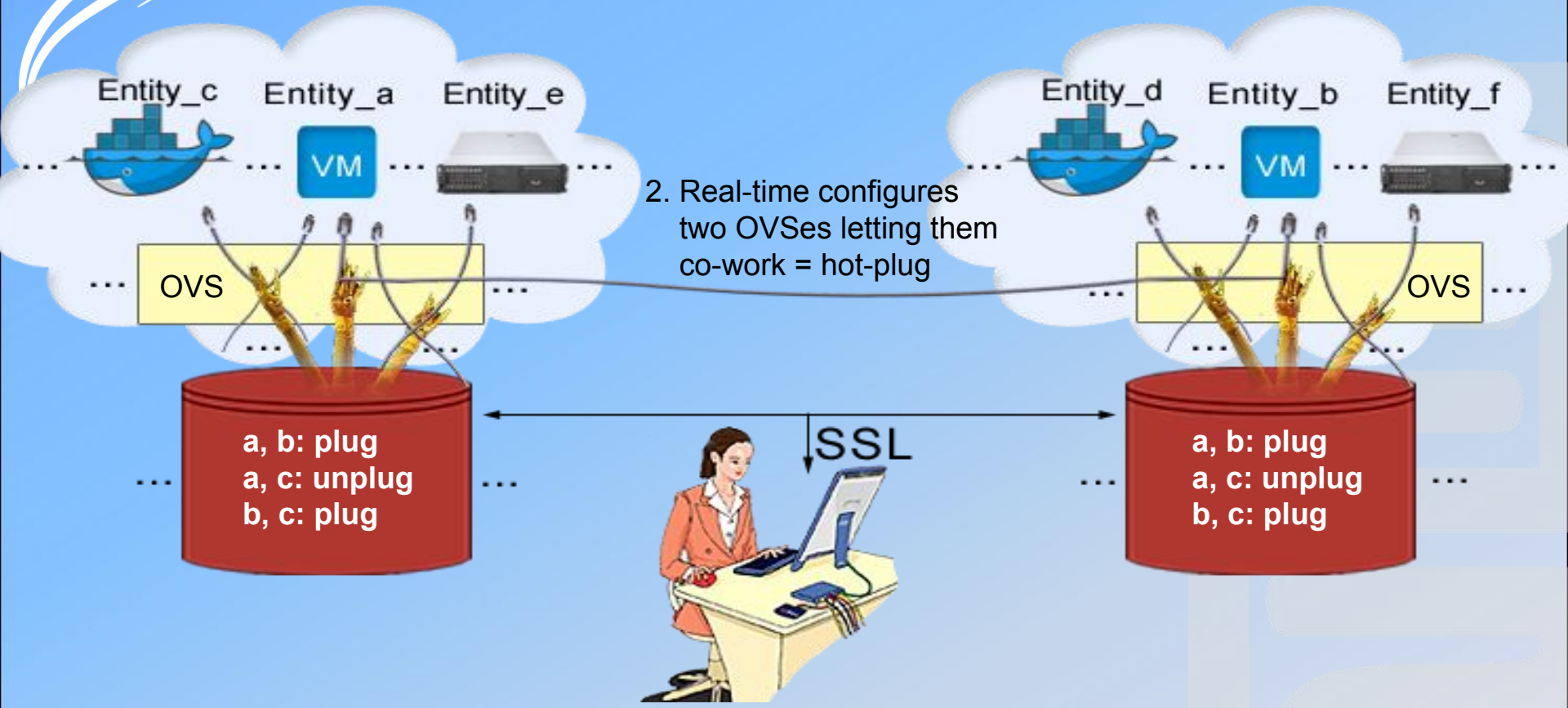
The route information is complete

The two routers do not learn, update, and maintain route tables; they are configured by the Controller in real time of container communication

No need of packet encapsulation

Overlay/underlay packets L2/L3 header data form a complete route	

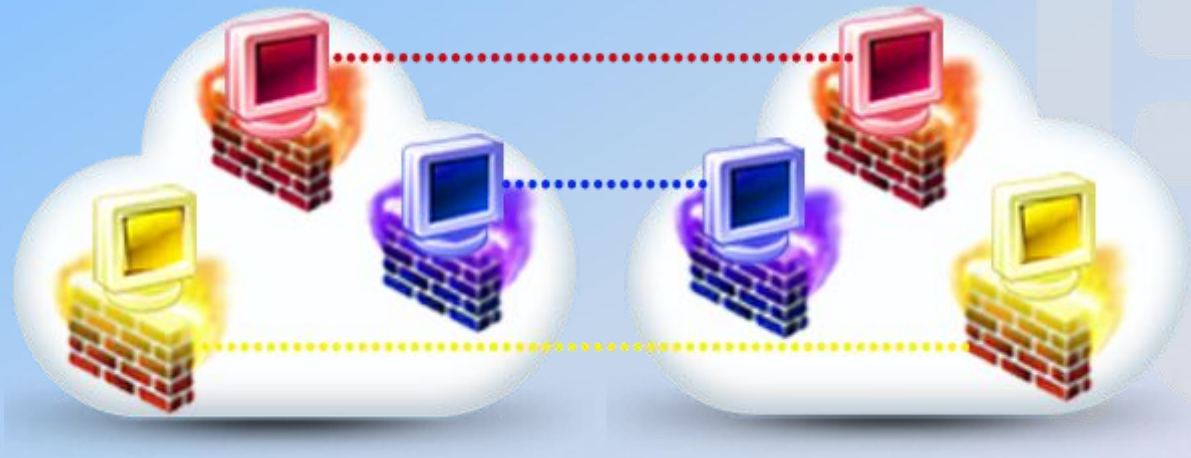
User-mode Connection of Overlay Network Lightweight and Simple to Manage



Controller only sees and handles the first packet as connection request
The remainder packets are shortcut directly between OVS servers

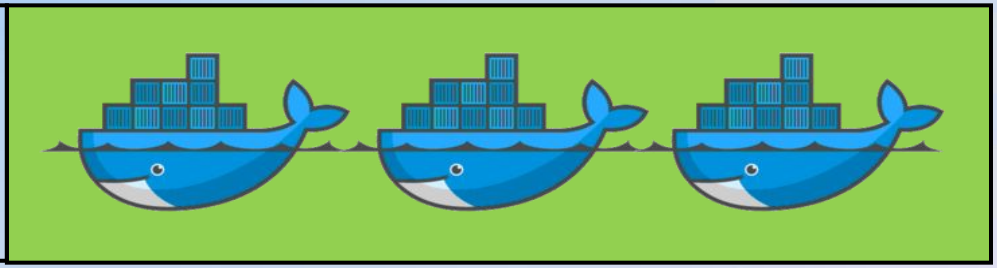
Improved Cloud Security

- Physically distributed controllers are only deployed in intranet, protected by firewalls
- SSL security protocol to protect (1) synchronization of network metadata, and (2) PacketIn and PacketOut flows (Openflow standard)
- Overlay nodes can talk one another only if the Controller PacketOut flows, thus, ARP attack using a wrong MAC address won't work, even for containers in one Docker server



Lightweight and Simple

- OVS never memorizes MAC addresses of containers on another OVS, no need of ARP flood, switching is purely enabled by PacketOut flow from Controller
- Controller only needs to record very few network metadata: MACs and IPs of containers, MACs and IPs of Docker servers, and security group information of containers
- No packet encapsulation, therefore no packet fragmentation, no nullification of networking diagnosing and trouble shooting tools, e.g., traceroute

Overlay/underlay packets L2/L3 header data form complete route	

OVS on Every Docker Server: Completely Distributed Switches, Routers and Gateways

- OVS in distributed Docker hosts form: vSwitch, vRouter, vGateway, vFirewall for Docker networking
- Containers within the same Docker server: if not allowed to connect, are isolated one another
- Containers can have usual intranet IP subnets: 10.0.0/8, 172.16.0/16, 192.168.0/24, OVS forming distributed vRouters forwarding between these subnets
- If a Docker host has external IP addresses, then OVS in the host forms a vGateway, and vFirewall, for containers to communicate with outside world

To Know More About DaoliNet

Open Source: github.com/daolinet

Demo System: www.daolicloud.com