# Issue

Typically, container-container networking issues manifest themselves as intermittent connection timeouts. They could also manifest as a complete failure of container-container networking.

This walkthrough will guide you to collect the necessary information to determine the root cause of a network connectivity problem.

# Prerequisites

First, determine the names of the affected Docker swarm services, networks, and hosts.

Identify a Docker host from which network requests are not answered correctly. If the requests are submitted through the ingress network, then to a frontend service, then via another network to a backend service, then start troubleshooting by splitting the problem in half and using `netshoot` to connect from the frontend directly to the backend service.

# Resolution

To troubleshoot container-container networking issues, start by examining the user-defined network. If the problem isn't found, move to troubleshooting the ingress network. Both are discussed below.

## Troubleshooting a User-Defined Network

1. On the host where the frontend container is running, start a netshoot container reusing the network namespace affected container:

   ```
   docker run -it --rm --network container:<^><container_name><^^> nicolaka/netshoot
   ```

   Steps 2 through 6 are executed inside this shell.

2. Look up all backend IP addresses by DNS name:

DNS names are created for containers and services and are scoped to each overlay network the container or service attaches to. Standalone containers user the container name for the hostnames. Looking up the name of a service returns the IP of the service's load balancing VIP. To lookup the IP of each task created by a service, use "task.<^>service_name<^^>" as the domain name.

For example, to lookup the IP addresses for a the backend service use:

~~~
```
nslookup tasks.<^>backend_service_name<^^>
```


> Where:
> - `<backend_service_name>` is changed to the backend service name
~~~

3. Issue a netcat TCP test to each backend task IP address on the port where it should be listening:

   ```
   nc -zvw2 $backend_ip <^><listening_port><^^>
   ```

> Where:
>
> - <listening_port> is changed to the port the backend tasks are listening on

To iterate over multiple task IPs for a service, you can use the following for loop:

```
for backend_ip in $(nslookup tasks.<^>backend_service_name<^^> 2>/dev/null \
  | awk '/answer:$/ {inanswer=1} inanswer && $1=="Address:" {print $NF}'); \
  do \
    nc -zw2  $backend_ip <^><listening_port><^^>; \
  done
```

Note: Output is only expected for IPs where connections fail.

5. If no connections fail but requests submitted via the ingress network continue to have problems, move to the next section on troubleshooting the ingress network. If no connections fail, and the issue has only been seen container-container, then check another set of services or hosts until one task fails to reach another.

6. For any backend IP addresses reported as failed, do a reverse name lookup of the affected IP to determine its service name and task id:

   ```
   nslookup <^><IP_address><^^>
   ```

   > Results will be formatted <^>service_name.slot.taskid.network_name<^^>

7. Exit the netshoot container and collect `docker network inspect -v` against the network between the two containers. Note the HostIP of tasks in the Services section that failed the netcat test.

8. On a manager, for the set of all failed service names and tasks, collect the following:

   ```
   docker service ps <^><service_name><^^>
   docker inspect --type task <^><task_id><^^>
   ```

9. For tasks that are still present (`inspect --type task` returns output), note their Created and Updated times. Collect Docker daemon logs covering this time frame from the netshoot host, all managers, and hosts of unresponsive tasks as identified by their HostIP.

   If your Docker daemon logs to journald, for example:

   ```
   journalctl -u docker --no-pager --since "YYYY-MM-DD 00:00" --until "YYYY-MM-DD 00:00"
   ```

10. Collect the output of https://github.com/docker/libnetwork/tree/master/support (https://github.com/docker/libnetwork/tree/master/support) from all hosts in the network path. This will expose and allow for the verification of the kernel programming if needed.

## Troubleshooting the Ingress Network

Service discovery is disabled on the ingress network for security reasons, so you can't use `nslookup tasks.service` to establish which backend IPs to test. Instead use the ipvs loadbalancer programming of the kernel.

1. On a manager, use `docker service inspect` to identify the VIP for the service on the ingress network (where <^><service_name><^^> is changed to the name of the service):

   ```
   ingress_id=$(docker network ls -qf name=ingress --no-trunc); docker service inspect <^>
   <service_name><^^> --format '{{range .Endpoint.VirtualIPs}}{{if eq "'${ingress_id}'" .NetworkID}}
   {{.Addr}}{{end}}{{end}}'
   ```

2. Using `curl` identify a Docker host from which network requests to the ingress published port of the service are not answered correctly:

   ```
   curl -4 -H 'Host: myapp.example.com' http://host:<^><listening_port><^^>
   ```

3. Use a netshoot container to enter into the ingress_sbox namespace on that host:

   ```
   docker run -it --rm -v /var/run/docker/netns:/netns --privileged=true nicolaka/netshoot nsenter --
   net=/netns/ingress_sbox sh
   ```

4. Look up the decimal ingress firewall mark for the service in question:

   ```
   iptables -nvL -t mangle |awk '/<^><vip><^^>/ {printf("%d", $NF)}'; echo
   ```

   > Where `<vip>` is replaced with the IP address of the service determined by service inspect.

5. List all the backend task IP addresses using ipvsadm

   ```
   ipvsadm -l -f <^><fwmark><^^>
   ```

   > Where `<fwmark>` is the decimal firewall mark extracted from the iptables mangle table.

6. Issue a netcat TCP test to each task IP address on the port where it should be listening:

   ```
   for backend_ip in $(ipvsadm -l -f <fwmark> | awk 'NR>3{print $2}' |cut -d: -f1); \
     do nc -zw1  $backend_ip <^><listening_port><^^> &>/dev/null || echo $backend_ip failed; \
     done
   ```

   > Where:
   > * `<listening_port>` is changed to the port the backend tasks are listening on
   > * `<fwmark>` is the changed to the decimal firewall mark extracted from the `iptables` mangle table

   Note: Output is only expected for IPs where connections fail.

7. Exit the netshoot container and collect `docker network inspect -v ingress`. Note the HostIP, service name, and task ID of any tasks with failed backend IP addresses.

8. On a manager, for the set of all failed service names and tasks, collect the following:

```
docker service ps <^><service_name><^^>
docker inspect --type task <^><task_id><^^>
```

9. For tasks that are still present (`inspect --type task` returns output), note their Created and Updated times. Collect Docker daemon logs covering this time frame from the netshoot host, all managers, and hosts of unresponsive tasks as identified by their HostIP.

   If your Docker daemon logs to journald, for example:

   ```
   journalctl -u docker --no-pager --since "YYYY-MM-DD 00:00" --until "YYYY-MM-DD 00:00"
   ```

10. Collect the output of https://github.com/adamancini/libnetwork/blob/improved_support_script/support.sh (https://github.com/adamancini/libnetwork/blob/improved_support_script/support.sh) from all hosts in the network path. This will expose and allow us to verify kernel programming if needed.