Container Networking

Glyn Normington Steve Powell

September 26, 2014

This document describes the container networking spec.

Contents

1 Introduction

This is a document that records the authors' deliberations as they come to grips with "what networking really does". It arises out of a need to control the networking of containers, including creating several containers which share a subnet, and the restriction of access of containers to the wider internet.

2 Overview of this document

This document is a place to store our initial thoughts as we investigate what the internet (IP) stack is like. The intention is to find the right decomposition of ideas to simply describe the state, and state transitions, of the components in the stack, and the tasks that they perform.

The document currently focuses on container networking and does not yet model the underlying IP and link layers. Behaviour in error cases is not specified; error cases are modelled simply as false preconditions.

The Z specification language is used to capture the model, but sufficient English text is also provided that readers who do not know Z should be able to understand the model. The appendix contains a summary of the Z notation.

3 Fundamentals

Nodes of the IP network are identified by IP address. Containers are identified by a container id. A subnet is a set of IP addresses. The formats of IP addresses and the structure of subnets need not be modelled for our current purposes.

$$[\mathit{IpAddr}, \mathit{ContainerId}]$$

$$\mathit{Subnet} == \mathbb{P}\ \mathit{IpAddr}$$

4 Network pool

A network pool is a managed collection of IP addresses which enables IP addresses to be allocated and freed. The pool is partitioned into allocated and free subsets.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{NetworkPool}} \\
\hline
\mathit{pool} : \mathbb{P}\, \mathit{IpAddr} \\
\mathit{alloc}, \mathit{free} : \mathbb{P}\, \mathit{IpAddr} \\
\hline
\mathit{free} \subseteq \mathit{pool} \\
\mathit{alloc} = \mathit{pool} \setminus \mathit{free} \\
\hline
\end{array}$$

5 Network pool operations

Operations on a network pool do not modify the pool of IP addresses.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{NetworkPoolChange}} \\
\hline
\Delta \mathit{NetworkPool} \\
\hline
\mathit{pool}' = \mathit{pool} \\
\hline
\end{array}$$

A single IP address can be allocated from the network pool.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Allocate}} \\
\hline
\mathit{NetworkPoolChange} \\
\mathit{ip}! : \mathit{IpAddr} \\
\hline
\mathit{ip}! \in \mathit{free} \\
\mathit{free}' = \mathit{free} \setminus \{\mathit{ip}!\} \\
\hline
\end{array}$$

An IP address and a gateway address can be allocated from the network pool.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{AllocateTwo}} \\
\hline
\mathit{NetworkPoolChange} \\
\mathit{ip}!, \mathit{gw}! : \mathit{IpAddr} \\
\hline
\mathit{ip}! \in \mathit{free} \land \mathit{gw}! \in \mathit{free} \\
\mathit{ip}! \neq \mathit{gw}! \\
\mathit{free}' = \mathit{free} \setminus \{\mathit{ip}!, \mathit{gw}!\} \\
\hline
\end{array}$$

The allocated addresses are distinct.

A set of IP addresses can be returned to the network pool. Any which did not come from the network pool are ignored.

$$
\begin{array}{|l}
\hline
\textit{Release} \\
\textit{NetworkPoolChange} \\
\textit{ips}? : \mathbb{P}\,\textit{IpAddr} \\
\hline
\textit{alloc}' = \textit{alloc} \setminus \textit{ips}? \\
\hline
\end{array}
$$

6 Container Network

A container network has an IP address, it belongs to a subnet, and it has a gateway address which it uses to route packets of data, known as *datagrams*, to non-local IP addresses. The container IP address identifies the container from a networking perspective as we will see later.

$$
\begin{array}{|l}
\hline
\textit{ContainerNet} \\
ip : \textit{IpAddr} \\
subnet : \textit{Subnet} \\
gw : \textit{IpAddr} \\
\hline
ip \in subnet \\
ip \neq gw \\
gw \notin subnet \\
\hline
\end{array}
$$

The container's IP address belongs to the container's subnet. The gateway address is distinct from the container's IP address and does not belong to the container's subnet.

6.1 Invariants

Each *ContainerNet* has a relationship with a particular *NetworkPool*. In practice, a network pool is associated with zero or more container networks.

CNetPooled holds when the container *ip* is not part of a special subnet, and is allocated from the pool.

$$
\begin{array}{|l}
\hline
\textit{CNetPooled} \\
\textit{ContainerNet} \\
\textit{NetworkPool} \\
\hline
ip \in pool \\
subnet = \{ip\} \\
gw \in alloc \\
\hline
\end{array}
$$

The container's gateway is allocated from the pool.

CNetSubnet holds when the container *ip* is part of a subnet potentially shared with other containers.

$$\begin{array}{|l}\hline \mathit{CNetSubnet} \\ \mathit{ContainerNet} \\ \mathit{NetworkPool} \\ \hline \mathit{subnet} \cap \mathit{pool} = \varnothing \\ \mathit{gw} \in \mathit{alloc} \\ \hline \end{array}$$

The container's gateway is allocated from the pool, but the container's IP address is not.

A container network is valid with respect to a network pool if either the container *ip* is allocated from the pool or is part of a special subnet.

$$\mathit{CNetValid} \mathrel{\widehat=} \mathit{CNetPooled} \lor \mathit{CNetSubnet}$$

There are also invariants between distinct pairs of *ContainerNet*s.

$$\begin{array}{l}\mathit{CNetPair} \mathrel{\widehat=} \mathit{ContainerNet}_1 \land \mathit{ContainerNet}_2 \\ \mathit{CNetDistinctPair} \mathrel{\widehat=} [\mathit{CNetPair} \mid \mathit{ip}_1 \neq \mathit{ip}_2]\end{array}$$

Any distinct pair of container networks have distinct container IP addresses since the container IP address identifies the container from a networking perspective.

Some distinct pairs of container networks share a special subnet.

$$\begin{array}{|l}\hline \mathit{CNetPairShared} \\ \mathit{CNetDistinctPair} \\ \hline \mathit{subnet}_1 = \mathit{subnet}_2 \\ \mathit{gw}_1 \neq \mathit{gw}_2 \\ \hline \end{array}$$

Such pairs of container networks have distinct gateway addresses.

Other distinct pairs of container networks have disjoint subnets. It turns out that at most one of the pair has a special subnet and at least one of the pair has a container IP address allocated from the network pool.

$$\begin{array}{|l}\hline \mathit{CNetPairDisjoint} \\ \mathit{CNetDistinctPair} \\ \hline \mathit{subnet}_1 \cap \mathit{subnet}_2 = \varnothing \\ \mathit{gw}_1 \neq \mathit{gw}_2 \\ \hline \end{array}$$

These pairs of container networks also have distinct gateway addresses.

A distinct pair of container networks is valid providing the pair shares a special subnet or they have disjoint subnets.

$$CNetPairValid \mathrel{\widehat{=}} CNetPairShared \lor CNetPairDisjoint$$

7 Network

The network, at least from the point of view of containers, has a network pool and a collection of container networks indexed by container id.

$$
\begin{array}{|l}
\hline
\textit{Net} \\
\textit{NetworkPool} \\
\textit{cnet} : \textit{ContainerId} \pfun \textit{ContainerNet} \\
\hline
\forall\, \textit{cn} : \operatorname{ran} \textit{cnet};\ \textit{ContainerNet} \mid \textit{cn} = \theta\textit{ContainerNet} \\
\qquad \bullet\ \textit{CNetValid} \\
\forall\, c_1, c_2 : \operatorname{dom} \textit{cnet};\ \textit{CNetPair} \\
\qquad \mid c_1 \neq c_2 \land \textit{cnet}\ c_1 = \theta\textit{ContainerNet}_1 \land \textit{cnet}\ c_2 = \theta\textit{ContainerNet}_2 \\
\qquad \bullet\ \textit{CNetPairValid} \\
\hline
\end{array}
$$

All the container networks in the collection are valid with respect to the network pool and with respect to each other.

8 Network operations

The following sections model operations on the network.

All operations on the network preserve the network pool of IP addresses (although IP addresses may be allocated and freed).

$$\textit{NetChange} \mathrel{\widehat=} \Delta\textit{Net} \land \textit{NetworkPoolChange}$$

8.1 Creating a container

We now specify what it means to create a container, from a networking perspective.

The basic principle of creating a container is that a new container id is generated and associated with a container network. Notice that the new container network is valid with respect to the network pool and with respect to any previously existing container networks.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{CNCreateBase}} \\
\hline
\textit{NetChange} \\
\textit{ContainerNet} \\
\\
\textit{cid}! : \textit{ContainerId} \\
\hline
\textit{cid}! \notin \operatorname{dom} \textit{cnet} \\
\textit{cnet}' = \textit{cnet} \cup \{\textit{cid}! \mapsto \theta \textit{ContainerNet}\} \\
\hline
\end{array}$$

There are three ways of creating a container network: the container IP address may be allocated automatically from the network pool; a specific container IP address may be allocated from the network pool; or a container network may be created for a special subnet. These variants are distinguished syntactically by the input parameters and are modelled below as three separate operations.

Firstly, a container network may be created by allocating its container IP address from the network pool.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{CNCreateFromPool}} \\
\hline
\textit{CNCreateBase} \\
\hline
\textit{AllocateTwo}[\textit{ip}/\textit{ip}!, \textit{gw}/\textit{gw}!] \\
\textit{subnet} = \{\textit{ip}\} \\
\hline
\end{array}$$

The container's IP and gateway addresses are allocated from the network pool. The container's subnet consists of the container's IP address alone.

Secondly, a container network may be created by allocating a container IP address specified as an input parameter from the network pool.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{CNCreateFromPoolWithSpecificIP}} \\
\hline
\textit{CNCreateFromPool} \\
\textit{ip}? : \textit{IpAddr} \\
\hline
\textit{ip}? = \textit{ip} \\
\hline
\end{array}$$

The specified IP address must be free for the operation to succeed.

Thirdly, a container network may be allocated with a special subnet and a container IP address from that subnet. This variant enables multiple containers to be created which share a subnet.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{CNCreateSubnet}} \\
\textit{CNCreateBase} \\
\textit{subnet}? : \textit{Subnet} \\
\textit{ip}? : \textit{IpAddr} \\
\hline
\textit{ip}? \in \textit{subnet}? \\
\textit{ip}? = \textit{ip} \\
\textit{subnet}? = \textit{subnet} \\
\textit{Allocate}[\textit{gw}/\textit{ip}!] \\
\hline
\end{array}
$$

The gateway address is allocated from the network pool. The invariants in *Net* imply that the specified subnet is either new or is identical to that of one or more existing containers in which case those containers have container IP addresses distinct from the specified container IP address. In other words, it is an error to specify a subnet and a container IP address that are already in use.

8.2 Destroying a container

Destroying a container from a network perspective simply means removing it from the collection of containers and releasing the IP addresses that were allocated for the container from the network pool.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{CNDestroy}} \\
\textit{NetChange} \\
\textit{cid}? : \textit{ContainerId} \\
\hline
\exists \textit{ContainerNet} \mid \textit{cid}? \mapsto \theta \textit{ContainerNet} \in \textit{cnet} \bullet \\
\quad\quad \textit{cnet}' = \{\textit{cid}?\} \ntriangleleft \textit{cnet} \land \\
\quad\quad (\exists \textit{ips} : \mathbb{P}\, \textit{IpAddr} \mid \textit{ips} = \{\textit{ip}, \textit{gw}\} \bullet \\
\quad\quad\quad\quad \textit{Release}[\textit{ips}/\textit{ips}?]) \\
\hline
\end{array}
$$

A Z Notation

Numbers:

$\mathbb{N}$	Natural numbers {0,1,...}

Propositional logic and the schema calculus:

$\ldots \land \ldots$	And	$\langle\!\langle \ldots \rangle\!\rangle$	Free type injection
$\ldots \lor \ldots$	Or	$[\ldots]$	Given sets
$\ldots \Rightarrow \ldots$	Implies	$'$, $?$, $!$, $_0 \ldots _9$	Schema decorations
$\forall .. \mid .. \bullet ..$	For all	$\ldots \vdash \ldots$	theorem
$\exists .. \mid .. \bullet ..$	There exists	$\theta \ldots$	Binding formation
$\ldots \setminus \ldots$	Hiding	$\lambda \ldots$	Function definition
$\ldots \widehat{=} \ldots$	Schema definition	$\mu \ldots$	Mu-expression
$\ldots == \ldots$	Abbreviation	$\Delta \ldots$	State change
$\ldots ::= \ldots \mid \ldots$	Free type definition	$\Xi \ldots$	Invariant state change

Sets and sequences:

$\{\ldots\}$	Set	$\ldots \setminus \ldots$	Set difference
$\{.. \mid .. \bullet ..\}$	Set comprehension	$\bigcup \ldots$	Distributed union
$\mathbb{P} \ldots$	Set of subsets of	$\# \ldots$	Cardinality
$\varnothing$	Empty set	$\ldots \subseteq \ldots$	Subset
$\ldots \times \ldots$	Cartesian product	$\ldots \subset \ldots$	Proper subset
$\ldots \in \ldots$	Set membership	$\ldots$ partition $\ldots$	Set partition
$\ldots \notin \ldots$	Set non-membership	seq	Sequences
$\ldots \cup \ldots$	Union	$\langle \ldots \rangle$	Sequence
$\ldots \cap \ldots$	Intersection	disjoint $\ldots$	Disjoint sequence of sets

Functions and relations:

$\ldots \leftrightarrow \ldots$	Relation	$\ldots^{*}$	Reflexive-transitive closure
$\ldots \nrightarrow \ldots$	Partial function	$\ldots ⦇ \ldots ⦈$	Relational image
$\ldots \rightarrow \ldots$	Total function	$\ldots \oplus \ldots$	Functional overriding
$\ldots ⤔ \ldots$	Partial injection	$\ldots \lhd \ldots$	Domain restriction
$\ldots \rightarrowtail \ldots$	Injection	$\ldots \rhd \ldots$	Range restriction
dom $\ldots$	Domain	$\ldots ⩤ \ldots$	Domain subtraction
ran $\ldots$	Range	$\ldots ⩥ \ldots$	Range subtraction
$\ldots \mapsto \ldots$	maplet		
$\ldots^{\sim}$	Relational inverse		

Axiomatic descriptions:

Declarations

Predicates

Schema definitions:

SchemaName

Declaration

Predicates