Container networking

Glyn Normington Steve Powell

September 25, 2014

This document describes the container networking spec.

Contents

1 Introduction

This is a document that records the deliberations of Glyn and Steve 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.

3 Fundamentals

$$[IpAddr, ContainerId]$$

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

4 Network states

Explain

ContainerNet

$$ip : IpAddr$$
$$subnet : Subnet$$
$$gw : IpAddr$$

$$ip \in subnet$$
$$ip \neq gw$$
$$gw \notin subnet$$

NetworkPool

$$pool : \mathbb{P}\ IpAddr$$
$$alloc, free : \mathbb{P}\ IpAddr$$

$$free \subseteq pool$$
$$alloc = pool \setminus free$$

5 Network pool operations

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

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

5.1 Invariants

Each *ContainerNet* has a relationship with the *NetworkPool*. *CNetPooled* holds when the container *ip* is not part of a special subnet, and is allocated from the pool.

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

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{CNetSubnet}} \\
\hline
\textit{ContainerNet} \\
\textit{NetworkPool} \\
\hline
\textit{subnet} \cap \textit{pool} = \varnothing \\
\textit{gw} \in \textit{alloc} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{CNetPooled}} \\
\hline
\textit{ContainerNet} \\
\textit{NetworkPool} \\
\hline
\textit{ip} \in \textit{pool} \\
\textit{subnet} = \{\textit{ip}\} \\
\textit{gw} \in \textit{alloc} \\
\hline
\end{array}
$$

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

We also have invariants that hold between pairs of *ContainerNet*s.

$$CNetPair \mathrel{\widehat=} ContainerNet_1 \land ContainerNet_2$$
$$CNetDistinctPair \mathrel{\widehat=} [CNetPair \mid ip_1 \neq ip_2]$$

CNetPairShared

$$CNetDistinctPair$$

$$subnet_1 = subnet_2$$
$$gw_1 \neq gw_2$$

CNetPairDisjoint

$$CNetDistinctPair$$

$$subnet_1 \cap subnet_2 = \varnothing$$
$$gw_1 \neq gw_2$$

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

6 More network state

Net

$$NetworkPool$$
$$cnet : ContainerId \rightarrowtail\!\!\!\!\!+ ContainerNet$$

$$\forall cn : \operatorname{ran} cnet;\ ContainerNet \mid cn = \theta ContainerNet \bullet CNetValid$$
$$\forall c_1, c_2 : \operatorname{dom} cnet;\ CNetPair \mid c_1 \neq c_2 \land cnet\ c_1 = \theta ContainerNet_1 \land cnet\ c_2 = \theta ContainerNet_2 \bullet CNetPairValid$$

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

7 Creating a container

$CNCreateBase$

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

$CNCreateFromPool$

$$
\begin{array}{|l}
CNCreateBase \\
\hline
Allocate[ip/ip!] \fatsemi Allocate[gw/ip!]
\end{array}
$$

A Z Notation

Numbers:

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

Propositional logic and the schema calculus:

$\ldots \land \ldots$	And
$\ldots \lor \ldots$	Or
$\ldots \Rightarrow \ldots$	Implies
$\forall .. \mid .. \bullet ..$	For all
$\exists .. \mid .. \bullet ..$	There exists
$\ldots \setminus \ldots$	Hiding
$\ldots \widehat{=} \ldots$	Schema definition
$\ldots == \ldots$	Abbreviation
$\ldots ::= \ldots \mid \ldots$	Free type definition
$\langle\!\langle \ldots \rangle\!\rangle$	Free type injection
$[\ldots]$	Given sets
$', ?, !, _{0 \ldots 9}$	Schema decorations
$\ldots \vdash \ldots$	theorem
$\theta \ldots$	Binding formation
$\lambda \ldots$	Function definition
$\mu \ldots$	Mu-expression
$\Delta \ldots$	State change
$\Xi \ldots$	Invariant state change

Sets and sequences:

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

Functions and relations:

$\ldots \leftrightarrow \ldots$	Relation
$\ldots \nrightarrow \ldots$	Partial function
$\ldots \rightarrow \ldots$	Total function
$\ldots \nrightarrowtail \ldots$	Partial injection
$\ldots \rightarrowtail \ldots$	Injection
$\mathrm{dom} \ldots$	Domain
$\mathrm{ran} \ldots$	Range
$\ldots \mapsto \ldots$	maplet
$\ldots^{\sim}$	Relational inverse
$\ldots^{*}$	Reflexive-transitive closure
$\ldots (\!| \ldots |\!)$	Relational image
$\ldots \oplus \ldots$	Functional overriding
$\ldots \lhd \ldots$	Domain restriction
$\ldots \rhd \ldots$	Range restriction
$\ldots \ntriangleleft \ldots$	Domain subtraction
$\ldots \ntriangleright \ldots$	Range subtraction

Axiomatic descriptions:

Declarations

Predicates

Schema definitions:

SchemaName

Declaration

Predicates

B References