



TrafficSqueezer and Aquarium

Series: Solar-Flare

User's Guide

- Book Version: 1.00.7

Author:

Kiran Kankipati

Author/Founder: TrafficSqueezer – Linux based Open-Source WAN Optimization

Author/Founder: Aquarium – Linux based Open-Source Web GUI

Author/Founder: CDN School – online CDN Technology Journal

Founder/CEO/MD: Doublefish Solutions

Disclaimer

By downloading and / or using these items you accept the terms of the basic License Agreement as shown on this document, and any additional terms in the individual license agreement supplied with the item. You have also accepted the general Disclaimer, so make sure you have read both before downloading or using anything from this section of the documentation.

Warranty Disclaimer. The software is provided to you "as is" without warranty of any kind, whether express, implied, statutory, or otherwise. KIRAN KANKIPATI AND ITS LICENSORS BEAR NO LIABILITY FOR ANY DAMAGES RESULTING FROM USE (OR ATTEMPTED USE) OF THIS PRODUCT/SOLUTION.

Author's Note and Project History

Dear All,

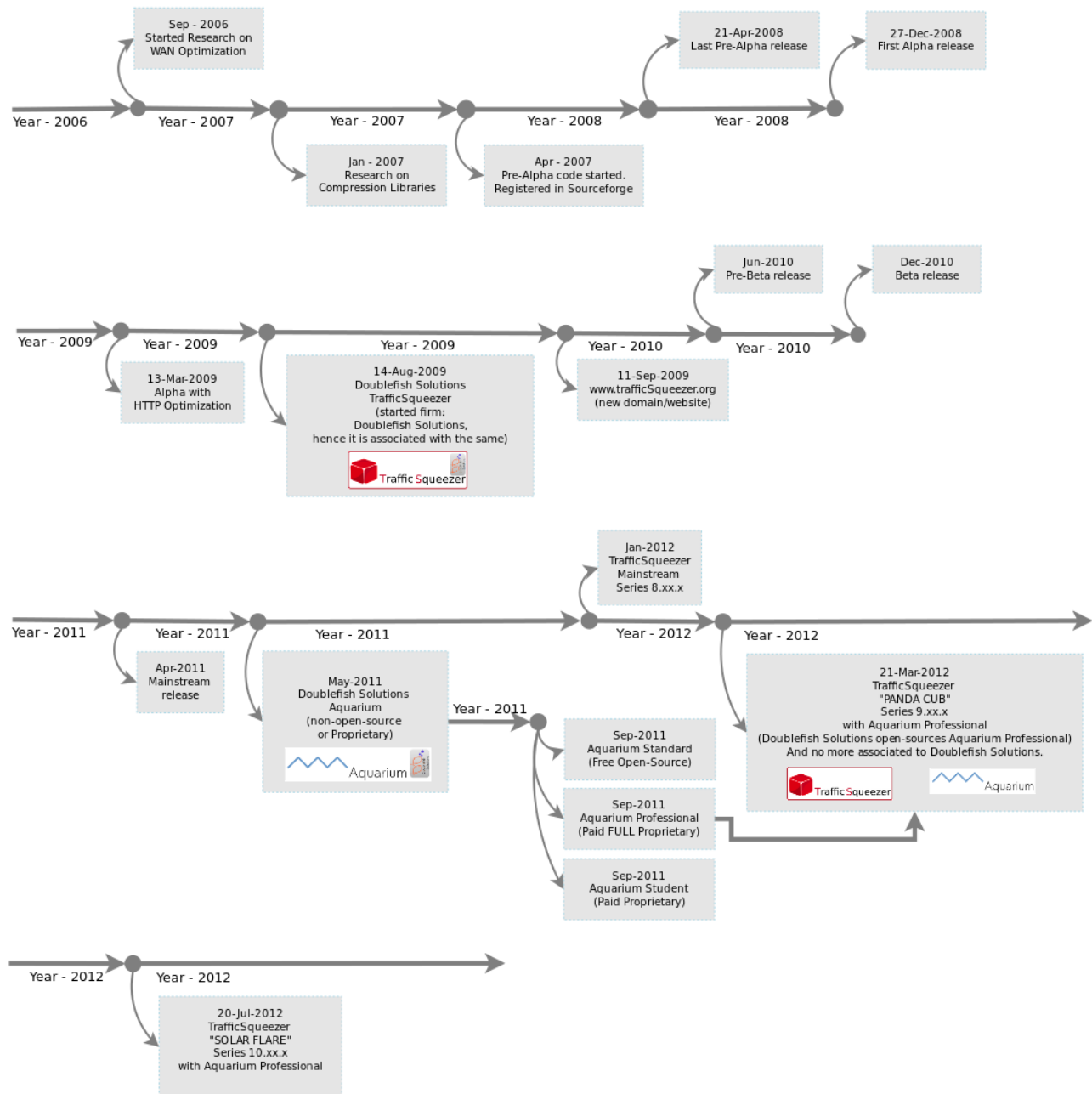
This is Kiran.

TrafficSqueezer is a project which was started as more of a hobby/fun project at home sometime during 2006. Since then with consistent R&D, it has been built from scratch in various phases. Initially a basic prototype of TrafficSqueezer is all built in user-space and demonstrated. Later the same is moved into Linux Kernel and a Kernel based practical usable WAN Optimization engines of TrafficSqueezer are born.

After few years, when I started my firm "Doublefish Solutions", TrafficSqueezer is supported via the same. Users are offered for the first time, paid support, porting and customization services. Also during this period a more robust TrafficSqueezer Device GUI (non-open-source) is made, which later eventually transformed into another stand-alone Open-Source project "Aquarium". Aquarium is a general purpose web-based GUI for Linux which also provides the GUI for TrafficSqueezer's Kernel core engines. Hence it is called "Aquarium", and it is a stand-alone project by itself. For various business reasons Aquarium partly made as open-source and partly (i.e a full feature version) is made as proprietary offering via my firm.

Due to various reasons, I did a closure of my firm "Doublefish Solutions", and so now once again TrafficSqueezer is solely maintained by me, and since being the sole Author/Founder of Aquarium, even Aquarium is now being solely maintained by me.

Here is a detailed historical road-map time-line picture:



I wish to thank all the supporters, users, and all my customers (during the tenure of Doublefish Solutions), for their interest, trust and support.

I got a great experience and fun in building/developing TrafficSqueezer. Sometimes it is like building a Rocket in my backyard. It is expensive, time-consuming, extremely complicated but yet at the same it is a great learning experience, to start something like this from scratch and develop further.

Besides TrafficSqueezer and Aquarium, here are my other areas of interest.

Open-Source: activist, promoter, contributor, Founder/Author: TrafficSqueezer , Aquarium
Networking: specialist, expert, protocol analysis, network hacking and researcher
Linux/Kernel: Linux Kernel Network flow specialist, Kernel developer, and reviewer
Business: analyst, market analyst and business strategist, consultant, and stock market analyst
Hobbies: painting, playing music, hobby cinematography and photography, writing articles
- cars and jeeps.
- Personal blog and updates - Kiran Kankipati: kirankankipati.blogspot.com
- Blog: Maruti Gypsy - Everything About: marutigypsy.blogspot.com



Thank you, Cheers, Kiran

License

TrafficSqueezer License:

TRAFFIC SQUEEZER provides dual licenses, designed to meet the usage and distribution requirements of different types of users.

GPLv2 License:

Copyright © (2006-2012) Kiran Kankipati, All Rights Reserved.
kiran.kankipati@trafficsqueezer.org
kiran_cyberpro@yahoo.com
kiran.cyberpro@gmail.com

Traffic Squeezer is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License Version 2, and not any other version, as published by the Free Software Foundation. TrafficSqueezer is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with TrafficSqueezer; see the file COPYING. If not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

BSD License (2-clause license):

Copyright © (2006-2012) Kiran Kankipati. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY KIRAN KANKIPATI ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL KIRAN KANKIPATI OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of Kiran Kankipati.

* This license is applicable exclusive to the Traffic Squeezer components. TrafficSqueezer may bundle or include other third-party open-source components. Kindly refer these component README and COPYRIGHT/LICENSE documents, released by its respective authors and project/module owners.

** For more details about Third-party components, you can also kindly refer TrafficSqueezer project website About page.

Aquarium License:

AQUARIUM provides dual licenses, designed to meet the usage and distribution requirements of different types of users.

GPLv2 License:

Copyright (C) (2011-2012) Kiran Kankipati , All Rights Reserved.
kiran.kankipati@trafficsqueezer.org
kiran_cyberpro@yahoo.com
kiran.cyberpro@gmail.com

Aquarium is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License Version 2, and not any other version, as published by the Free Software Foundation. Aquarium is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with Aquarium; see the file COPYING. If not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

BSD License (2-clause license):

Copyright (2011-2012) Kiran Kankipati. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY KIRAN KANKIPATI ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL KIRAN KANKIPATI OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of Kiran Kankipati.

Introduction

Aquarium is a free Open-Source web-GUI (Graphical User Interface) for Linux based systems.

Aquarium is focused uniquely for deployments such as IaaS (Infrastructure as a Service), SaaS (Software as a Service), Cloud Networks, Distributed Computing deployments, as a generic purpose Linux GUI, generic purpose Linux based networking device GUI, GUI for TrafficSqueezer, Squid, and so on.

Besides, the main other role of Aquarium is to also provide a high-level Basic-Level and Advance Level of GUI interface and Abstraction to the existing traditional Linux based services.

Aquarium is not a replacement for traditional Linux powerful CLI components. Instead Aquarium aims to provide GUI for Linux components so that it gets ***platform-independent unified integrated web-GUI*** !

For more information about Aquarium Dependencies, kindly refer the topic ***"Aquarium Dependencies"***

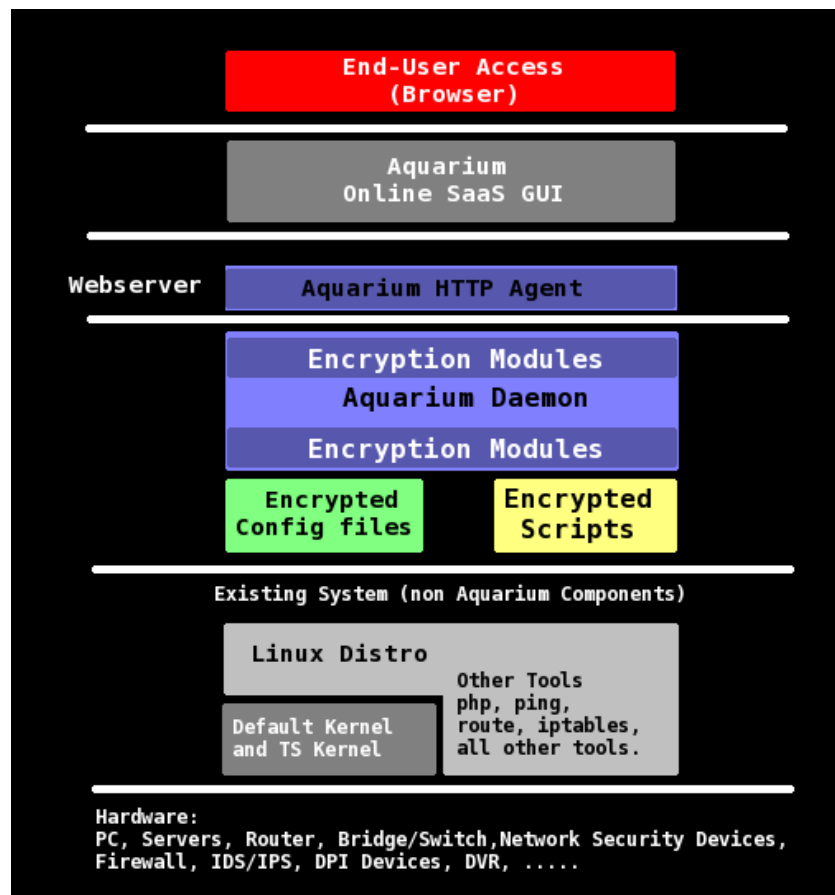
Aquarium - Architecture

Aquarium is not a replacement for traditional Linux interfaces namely

- remote SSH CLI shell access (or)
- local or remote (via VNC) GNOME/KDE window manager GUI access
- and so on ...

Instead Aquarium is a unique Linux web-GUI access to any supported Linux based systems.

Here is the picture which illustrates the complete Architecture of a typical SaaS installation based Aquarium end-to-end Architecture.



Aquarium - Dependencies

Loosely-coupled Architecture:

Aquarium Daemon and other components are highly based on LOOSELY-COUPLED ARCHITECTURE. It is so, since it is very important that Aquarium main components don't get crashed and don't get unwanted unnecessary long sequence of code. Hence it is the foremost priority that the code is maintained to be **loosely-coupled and crash tolerant** as much as possible.

Reliability:

The main Aquarium Daemon is written in C Language. It is so, since C-Language long-time execution mode programs are much more reliable. Versus a quick PHP/PERL based scripts are more suited for on-demand execution mode programs. They may often hit issues like memory leaks and other unpredictable issues, if the same program (technically same process i.e same process ID) executes for more than a week in that server. Although PHP-5 and PERL interpreters we have today had so much addressed these issues and limitations, still it is unpredictable, hence Aquarium main Daemon is written in C.

Dependencies:

- PHP – Parts of Aquarium code is been written in PHP. This is a critical code, if they gets crashed in any case, this will not crash Aquarium Daemon. The C-based Aquarium code will decrypt and load on-demand.
- grep, sed, cut, arp, iptables, cron, route, tc, gpg, egrep, cat, ifconfig, etc - Aquarium also uses these tools and may call them via exec() or system() APIs. Also may call dynamic bash scripts cooked (created) via Aquarium C-Language Daemon.

- Other third-party Libraries – Aquarium Daemon does multi-threading, hence it is dependent on the run-time pthread library, and other very basic default Linux components.
- MySQL – Aquarium stores all Aquarium GUI themes, and other component Statistics and Logs via MySQL DB tables. Hence it is mandatory that the system contains MySQL pre-installed and pre-configured (this documentation contains simple steps how you can create the same).
- Apache – Aquarium GUI will be installed in the web-root of Apache webserver, so that you can get the GUI access via local web-server on to a browser. Hence it is mandatory that the Apache is pre-installed and pre-configured.

Aquarium – Abstraction Features

Aquarium tries to provide Abstraction (a basic or advance GUI also if required an abstracted GUI) for many Linux components/modules.

For example, here is an abstracted GUI for configuring Linux static routing tables. As you can see the screen-shot, it is an abstraction of what users can do via a Linux **route** command.

route CLI Linux command vs. Aquarium Abstracted GUI ?

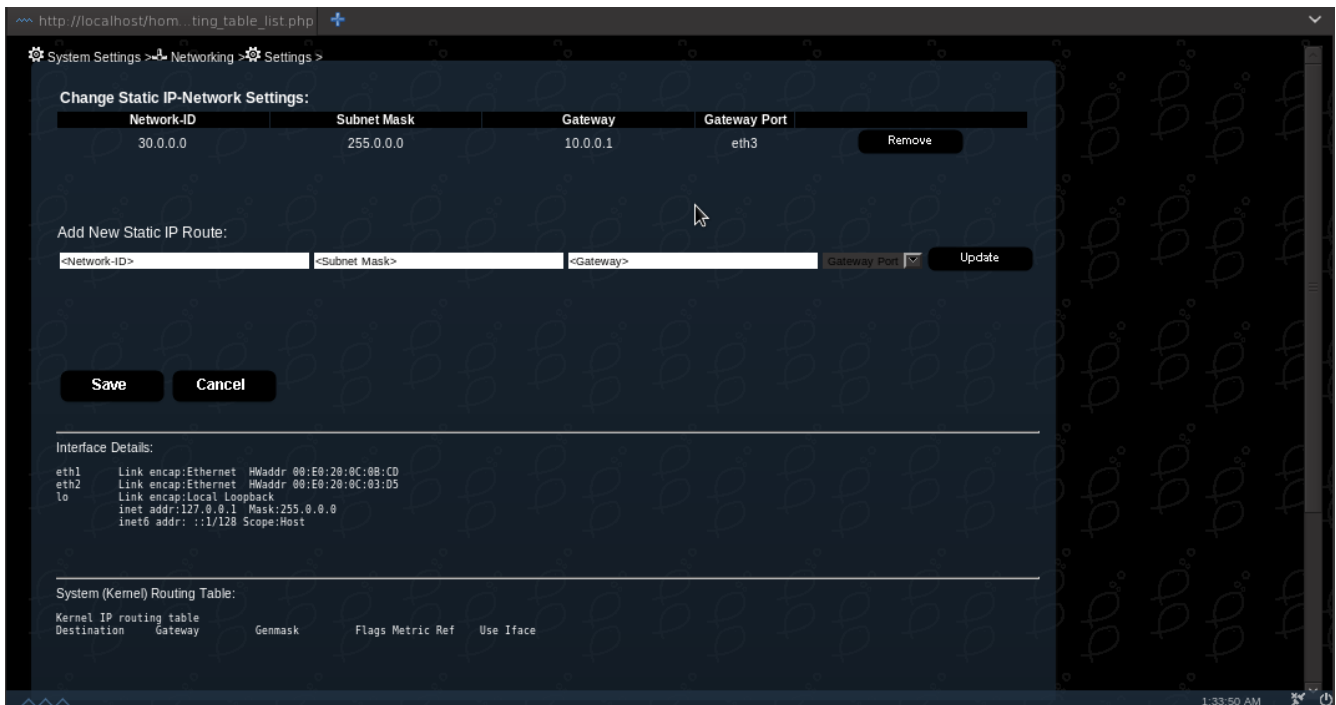
A CLI route command is highly versatile and very powerful as we are well aware. So Aquarium will provide one or many abstracted GUI for the same.

But why many Abstracted GUI pages ?

Actually sometimes the same CLI command, you can configure one or many different scenarios. You may also use various optional parameters based on your requirement. This is what exactly classified in Aquarium, and based on each specific scenario, it is been provided as an abstracted GUI via Aquarium.

This is the reason Aquarium is not a CLI replacement, it is meant to mainly provide abstraction for existing CLI command structure.

So the main focus of Aquarium is to provide simple unified integrated Interface !

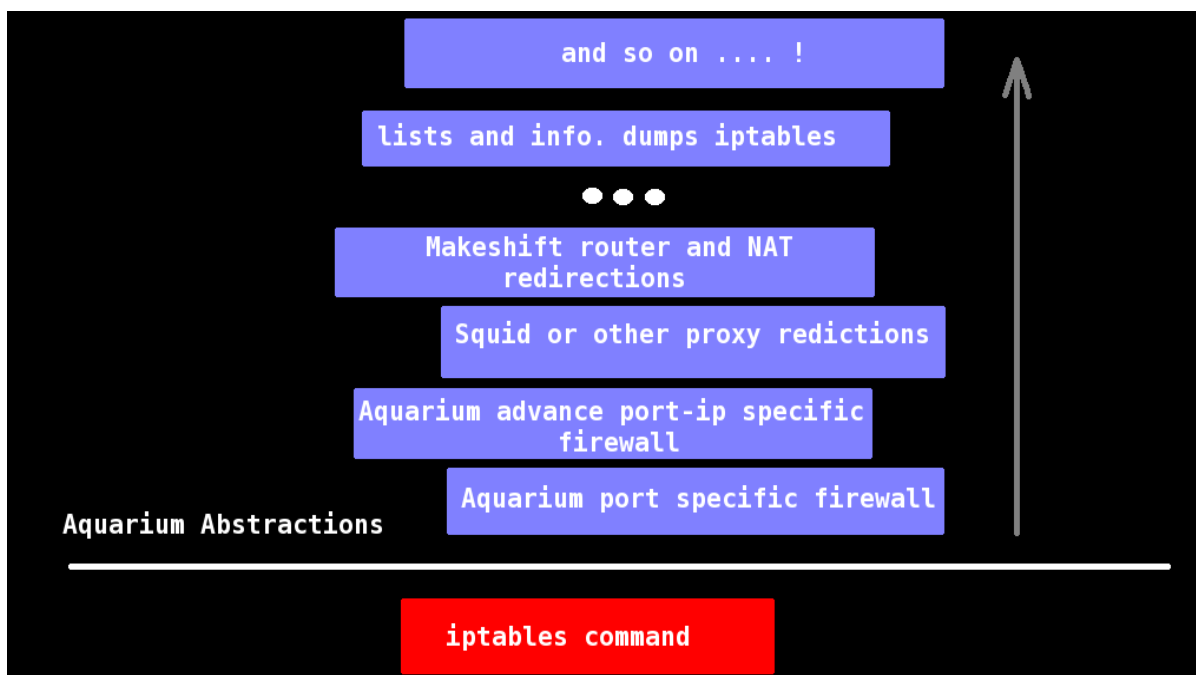


As you can see above screen-shot, Aquarium will not completely hide/abstract the user aware CLI command structure. Instead, it provides GUI plus also hints the respective actual device settings via context specific CLI command output/dumps.

Aquarium will execute these commands (in the above case as you can see it had executed `ifconfig` and parsed output, and `route` command) then if required it will capture the output and shows the same in the corresponding page.

This abstracted Aquarium interface is also can be explained as below:

For example if Aquarium has a GUI (pages) for Linux iptables (firewall) feature, then it may abstract many times as shown below:



As you can see based on each abstraction, the same underlying feature is or can be used any number of times in various contexts.

Hence this is the reason Aquarium is focused towards basic and advance users, where Aquarium really helps administrators to remotely and quickly administrate their systems in the usual three work modes.

Usually administrators can work in 3-work modes:

- **System Monitoring** – Administrators usually work in this mode everyday, once they first immediately start their shifts. They would like to monitor first everything is in the flow. Also sometimes middle of the day they may work in this mode.
- **Crisis and or Firefighting mode** – as the name explains this is another mode when there is a huge crisis or some business impacting issue found, then they work in this mode. Aquarium can be used initially to debug the system, even before explicitly administrators do and control via any SSH access. Sometimes SSH access can be eliminated via Aquarium Abstractions !

Administrators can connect to their devices via Aquarium (if not present locally), via any devices, mobile phones, iPod, or any device and can immediately take a swift control of the situation.

- **Casual Routine Maintenance mode** – This is again a unique mode, as the name suggests, sometimes if administrators want to perform a quick small task, then they can immediately do the same via Aquarium.

Hence these are various factors considered during in-terms of design of Aquarium platform and its road-map.

There is also a consistent effort that under no circumstances Aquarium platform will not get into a re-invent the same wheel scenarios. If not always, but atleast as much as possible it is prevented.

Aquarium – Current Compatibility – and other specifications

Browser:

Aquarium is 100% compatible with Mozilla Firefox Browser. Besides it is also mostly compatible with respect to other Mozilla-browser based platforms and others namely Opera, Google Chrome, Apple Safari, Midori, and so on. Aquarium is partially compatible with Microsoft Internet Explorer (IE), one of the reasons it is partially compatible is for the reason that IE does not support all specifications of browser CSS styling standards and web-kit standards.

Aquarium GUI components has less java-script intensive scripts, but done mostly with highly reliable server side light-weight PHP scripts and styled exclusively with CSS property styling.

Linux Distribution(s):

Currently Aquarium Professional is 100% compatible with distributions such as:

- Fedora Core (preferably Fedora Core 17)
- CentOS

Aquarium – Current Supported Features

Currently Supported:

Aquarium currently supports these listed features:

- 100% Doublefish Solutions TrafficSqueezer – Open-Source WAN Optimization Solution GUI.
 - WAN Optimization modules – settings, management, statistics, etc.
 - DPI modules
 - L7-filter modules

- TrafficSqueezer Bird's Eye view.
- Other TrafficSqueezer features.
- TrafficSqueezer configuration wizard.
- Basic-level iptables firewall GUI.
- Grub bootloader GUI.
- Basic-level file-manager GUI.
- Basic-level image-viewer.
- Basic-level MySQL abstracted GUI.
- Basic-level Linux static ip-routing tables.
- Basic-level and integrated Linux QoS Abstraction.
- Networking tools – ping, traceroute, netstat, route, arp, dig, and many more.
- GUI for some important /proc read-only system monitoring files.
- GUI to manage init-scripts (start-up run-level execution mode daemons).
- Basic-level Linux system user management.
- Other system specific features.
- Basic-level Linux bridging mode setup (via TrafficSqueezer GUI).

Aquarium – Installation

Setup Aquarium Daemon and Agent:

To install Aquarium, first you need to compile the sources, as shown below:

First enter into the ../aquarium folder, then compile the same to get release build.

```
[root@baracuda aquarium]# php -f make_release.php
```

Now you need to install this release which is now built and ready. So follow these steps:
Enter/Change into the release folder.

```
[root@baracuda aquarium]# cd release/
```

Upon a successful build (last step), you can see all the files over here. So now you can run setup script to install.

```
[root@baracuda release]# ./setup
```

Welcome to Aquarium - setup wizard

```
Kindly select
[i]nstaLL/[I]nstaLL with no confirmation and [u]ninstall/[U]ninstall with no confirmation: [i/I/u/U]: I
Installing Aquarium ...
Stopping aquariumd:           [ OK ]
Stopping aquariumd:           [FAILED]
Starting aquariumd:           [ OK ]
Stopping crond:                [ OK ]
Starting crond:                [ OK ]
Starting httpd:                [ OK ]
Starting mysqld:               [ OK ]
Done !
Installing Aquarium Agent in (/var/www/html) Apache default webroot folder ...
Done !
```

Thank you !

```
[root@baracuda release]#
```

Now you are done with the Aquarium daemon and agent setup.

Setup Aquarium Web GUI files:

Now you can copy the Aquarium front-end web resources into your default Apache web-server web-root folder:

```
[root@baracuda aquarium]# cd saas_gui
[root@baracuda saas_gui]# ls
agent db_saas html
[root@baracuda saas_gui]# cd html/
[root@baracuda html]# cp -r * /var/www/html/.
```

MySQL Database Setup for Aquarium:

Aquarium maintains its limited meta-data, themes, partially some configurations, and historical statistics in MySQL based DataBase tables. It should also store important TrafficSqueezer historic statistics, Squid-Cache configurations, Squid-Cache statistics and so on.

Hence it is mandatory that users do MySQL setup too.

IMPORTANT NOTE: Aquarium may store tables/DBs in the name of Traffic Squeezer or something else, users can ignore this nomenclature which is used.

NOTE: However in future, Aquarium may remove this manual mode MySQL setup, and instead support the same via the integrated Aquarium setup utility itself.

Here is a simple step-by-step procedure to set it up:

Turn on mysqld service/daemon:

```
#service mysqld start
```

We create/configure MySQL for admin user 'root' and password 'welcome', here are the simple steps, which you can change them accordingly:

Test Mysql access:

```
#mysql -u root
```

Configure password for user root:

```
#mysqladmin -u root password 'welcome'
```

Test Mysql access with password:

```
#mysql -uroot -pwelcome
```

Create a Database:

```
mysql> create database aquarium;
```

Work within Database:

```
mysql> use aquarium;
```

Create one basic pre-install/setup Table:

```
mysql> CREATE TABLE profile (  
    id int(10) unsigned NOT NULL AUTO_INCREMENT,  
    profile varchar(200) NOT NULL DEFAULT 'NULL',  
    username varchar(30) NOT NULL DEFAULT 'NULL',  
    password varchar(30) NOT NULL DEFAULT 'NULL',  
    db varchar(30) NOT NULL DEFAULT 'NULL',  
    PRIMARY KEY (id)  
);
```

Now create one single row within this table:

```
insert into profile (id, username, password) values (1, "root","welcome");
```

IMPORTANT NOTE: This table (profile) is necessary so that you can login in the Aquarium for the first time.

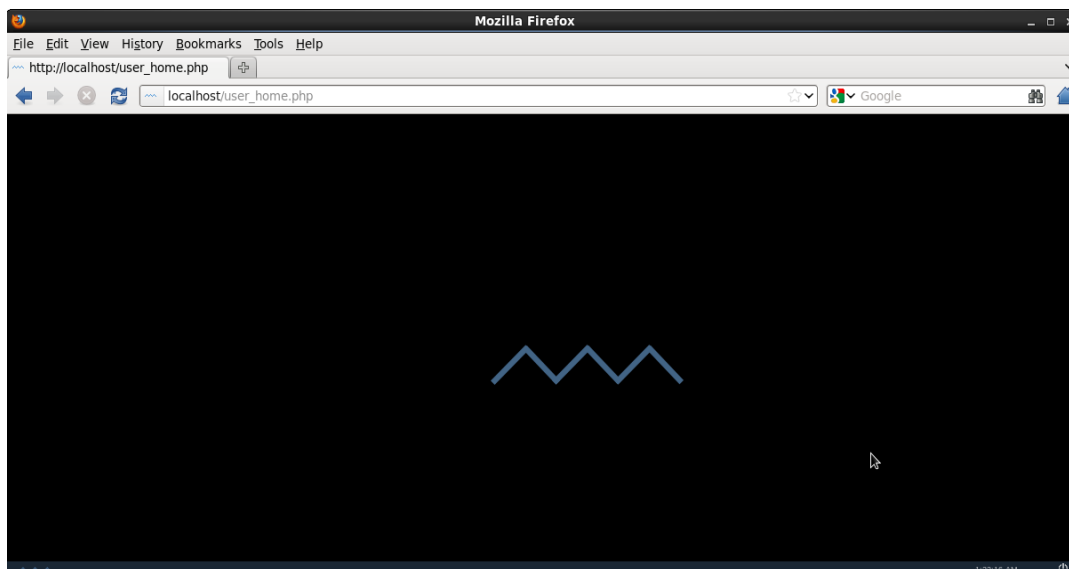
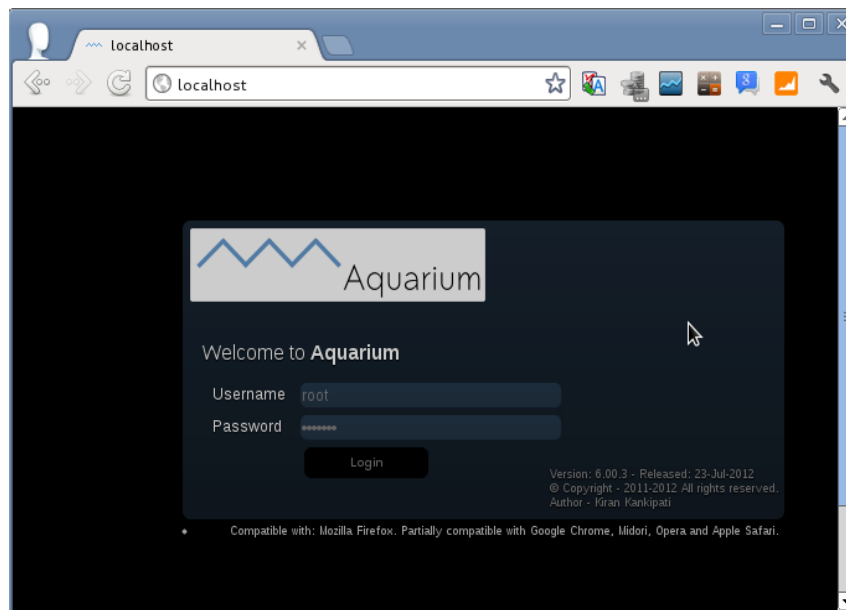
IMPORTANT NOTE: Do not change the Database password/user details. Aquarium is hard-coded to access local MySQL database with this access credentials. Instead you can block external users from accessing this database altogether via MySQL advance configuration.

Grant All permissions:

```
mysql> grant usage on *.* to root@localhost identified by 'welcome';
```

Aquarium – web-GUI

You can access on-line Aquarium GUI via your browser:

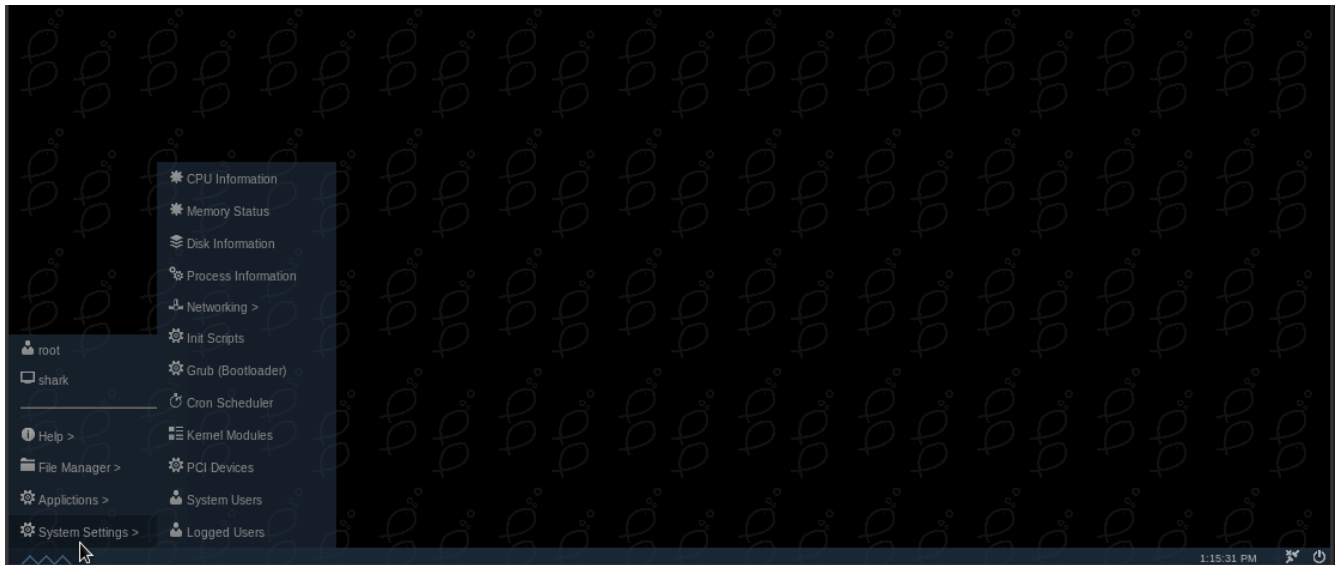


Basic Interface Layout:

Unlike a traditional web-site look and feel, Aquarium is deliberately made to look like any normal PC/Service GUI interface.

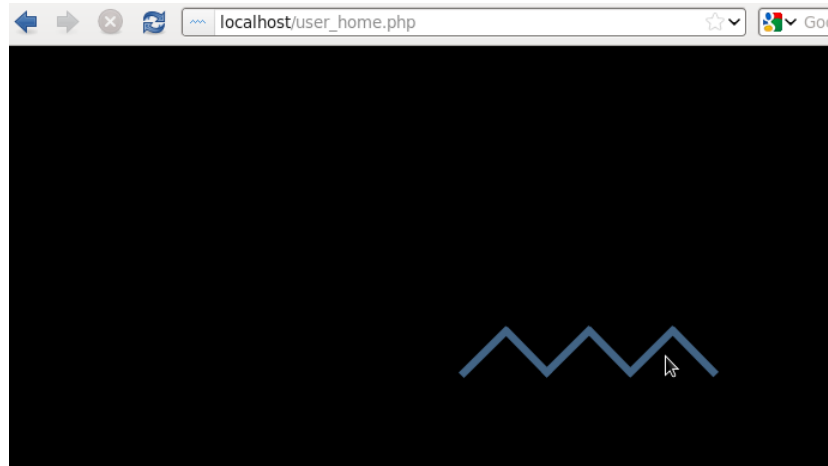
Navigation:

You can navigate most items via this below shown bottom taskbar->menus->submenus.

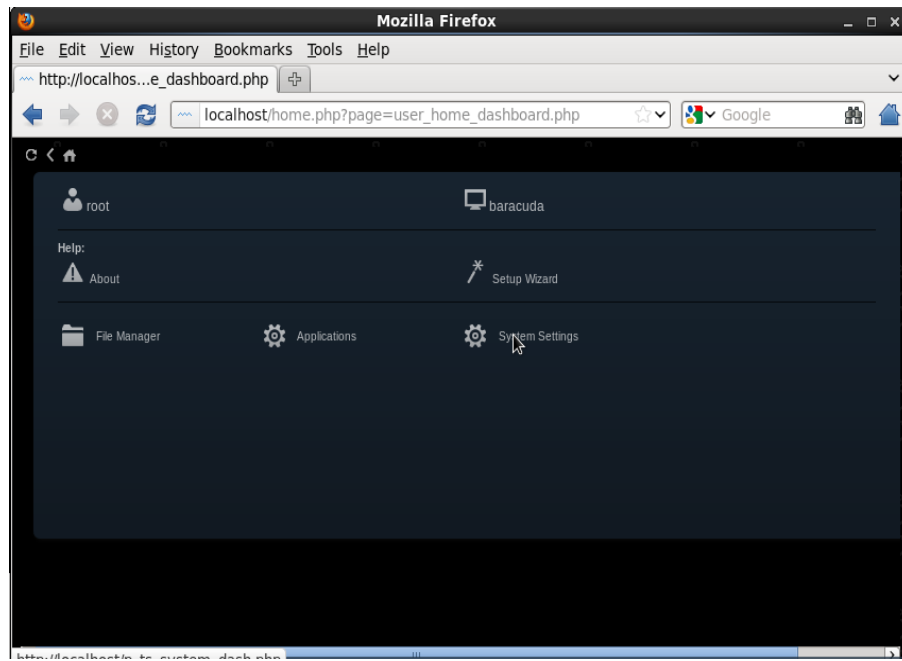


Navigation via Tablet PC and or via iPad:

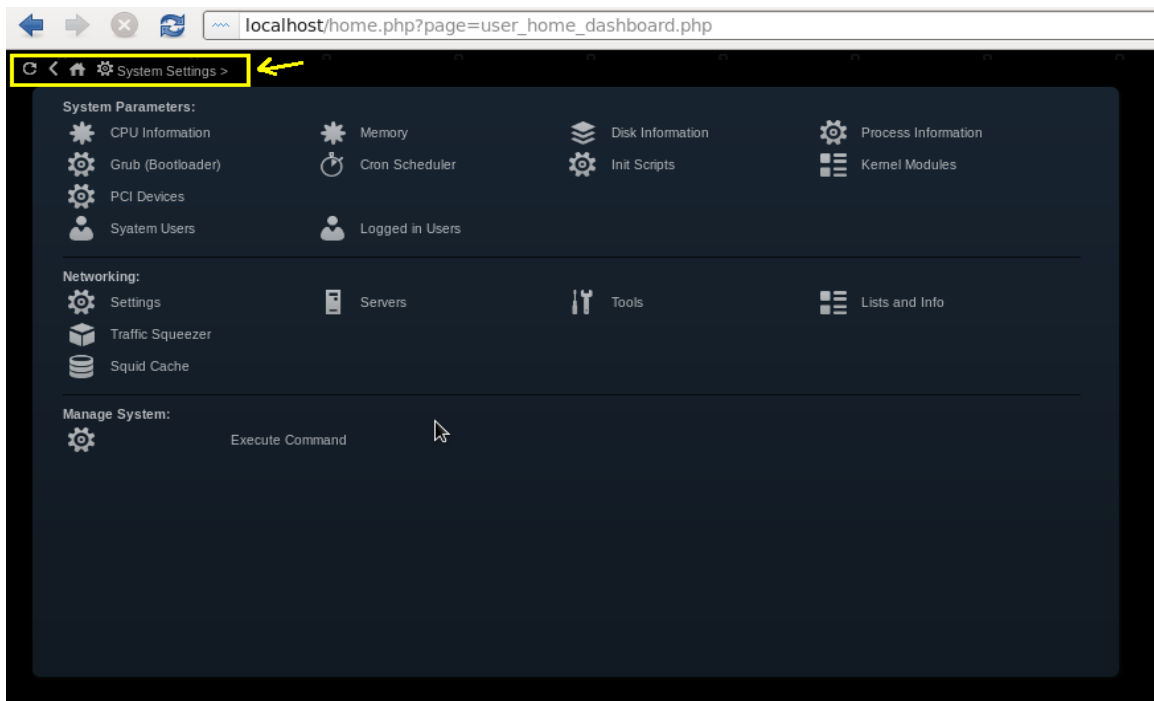
If users use Tablet PC and or iPad, and remotely connect the system which has Aquarium GUI, then they can use directly the logo of “Aquarium” in the middle to navigate through as shown in the below sequence.



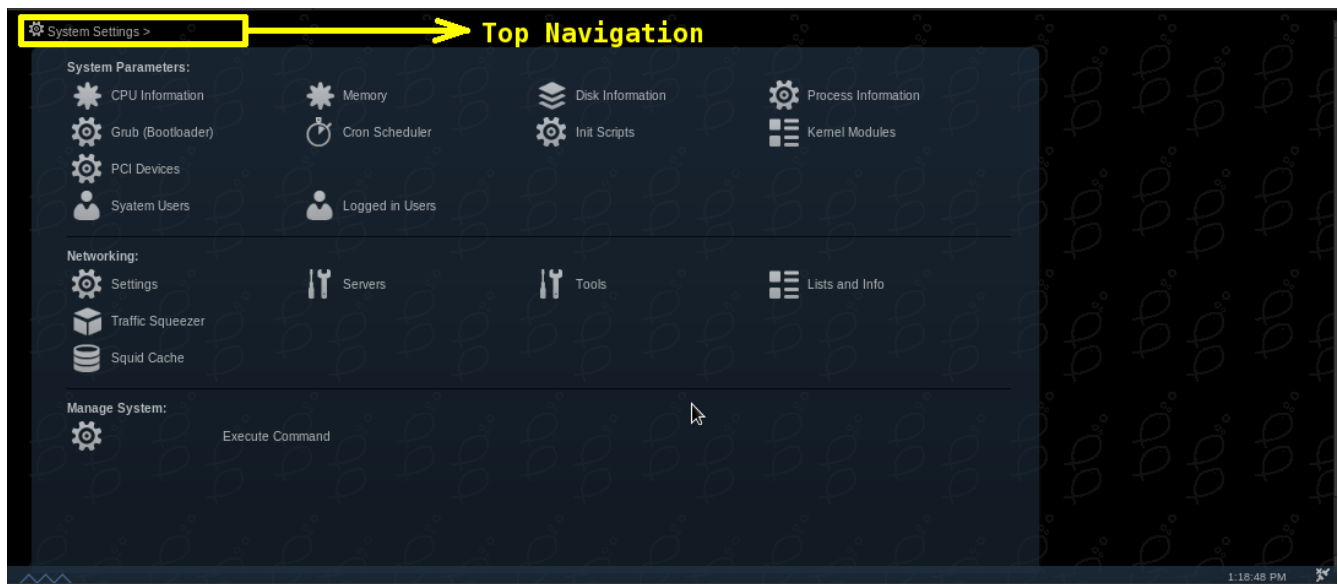
Upon clicking this middle Aquarium Logo, it opens the intermediate dashboard. And so you can navigate across all the portions of Aquarium. Hence this is an alternate way to use Aquarium without using its bottom taskbar.



Besides you should also get these intermediate dashboards with which you can navigate further below listed items.



Along with these you can also navigate each page via these top navigation entities.



For more information and for a detailed picture about Aquarium GUI kindly refer these YouTube videos where you can see live flow of the same rather than these screenshots.

NOTE: Your Aquarium version and the Aquarium in these videos might differ, but the basic look-and-feel and the concept should remain the same.

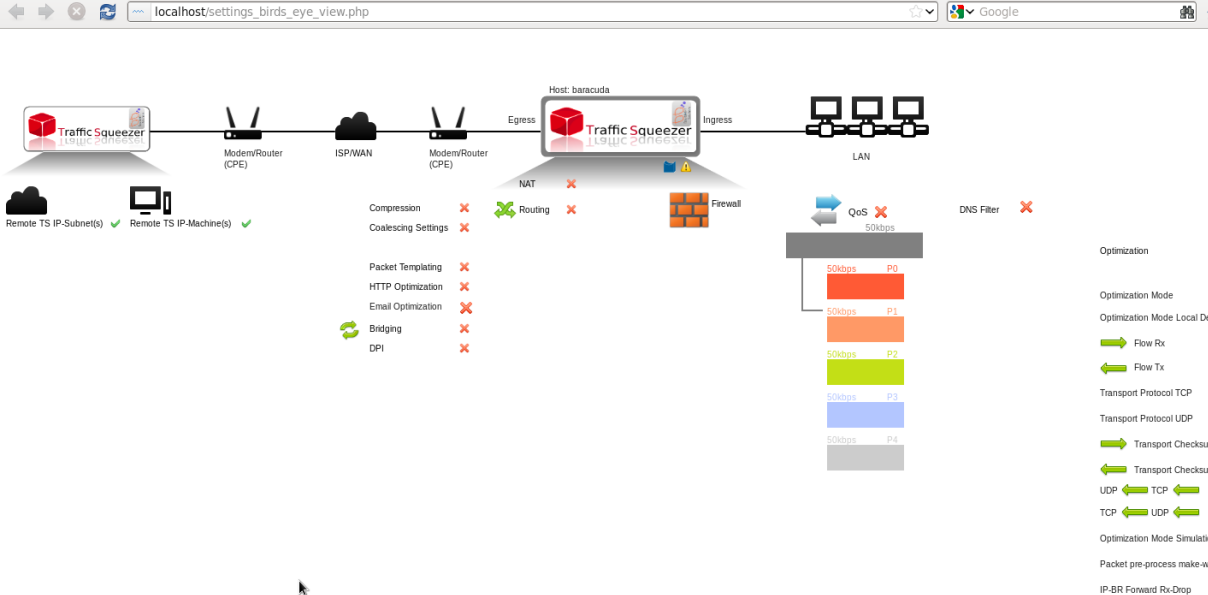
Aquarium Bird's Eye views:

For each major application and or open-source project Aquarium offers Bird's eye view for

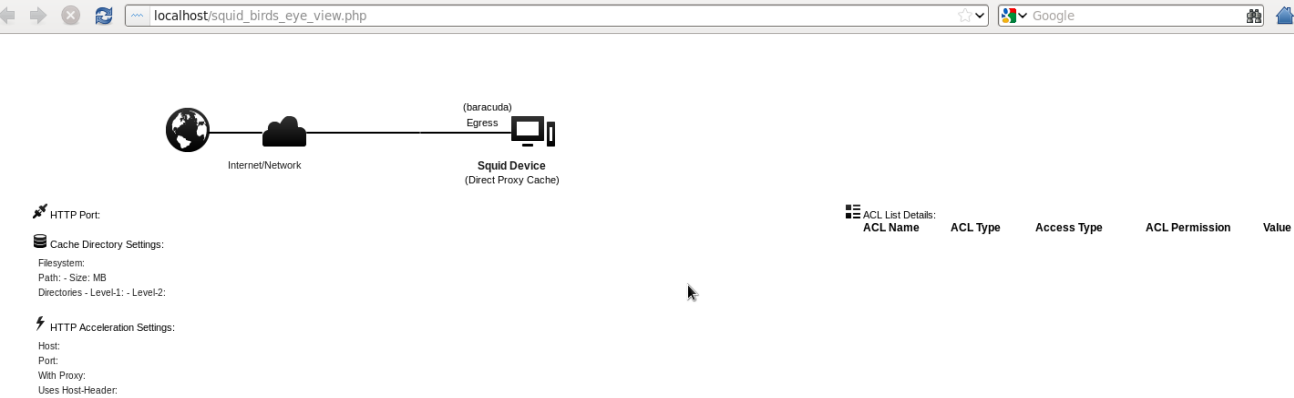
the same. This is like a generic “show” CLI command of most devices in GUI format. Bird's eye view shows the overall picture or scenario of your entire system settings of that feature/project/module/application.

So Aquarium offers bird's eye view for TrafficSqueezer and Squid-Cache software. The other main purpose and use of bird's eye view is that users (Administrators) can take print-out of these pages and if required can file them and even mark them on these papers for regular use and for offline reference. Also if required, Administrators can even paste these printed sheets on wall or near the server(s) for quick reference. And even in this case they can mark further any details directly on top of this printed sheet/chart.

Here is the sample screenshot of Aquarium bird's eye view of TrafficSqueezer:



Here is the sample screenshot of Aquarium bird's eye view of Squid:



Videos of Aquarium Demo:

Demonstration - Aquarium - 0.02.7

URL: <http://www.youtube.com/watch?v=cbqGqN8Gw9I>

Videos of other specific scenarios with Aquarium:

Demonstration of Traffic Squeezer - Single system simulation mode

URL: <http://www.youtube.com/watch?v=dT9hXYxtseA>

Demonstration of Traffic Squeezer - HTTP Access DPI Logs - with Aquarium

URL: <http://www.youtube.com/watch?v=yt8fxEDn0uo>

Traffic Squeezer - Real-time Benchmarking

URL: http://www.youtube.com/watch?v=kCnPqt71_6E

Demonstration - L7 filter - DNS Block feature via Aquarium

URL: <http://www.youtube.com/watch?v=huL8zmr5vkU>

In future you should get even more scenario specific Aquarium Demonstration videos, which should be also added into this document to refer specific sections or scenarios.

TrafficSqueezer Introduction

To read about generic introduction, theory and details about TrafficSqueezer, kindly refer the existing online documentation section from the official TrafficSqueezer website.

The main TrafficSqueezer Documentation section: <http://www.trafficsqueezer.org/doc.php>

NOTE: TrafficSqueezer online documentation should have minimal multi-language support. All the pages are translated via Google Translate tool. Hence users can partially get multi-language documentation.

First things First, what is the exact definition of TrafficSqueezer?

What is TrafficSqueezer?

According to me there are 3-4 definitions for TrafficSqueezer, and strictly it is not one solution, as many users assume. **If you assume TrafficSqueezer is for only WAN Optimization, then you are extremely WRONG!**

- **TrafficSqueezer is a WAN Optimization "SOLUTION".** - it has complete collection of modules/components & tools to use it as an end-user WAN Optimization solution.
- **TrafficSqueezer is a WAN Optimization "FRAME-WORK".** - it can be used as a frame-work and one can (mostly a firm or OEM), can build their WAN Optimization device. In this case TrafficSqueezer is a frame-work and raw material for this job/requirement.
- **TrafficSqueezer is a network Packet Engineering solution.** - forget WAN Optimization, if you want to custom engineer network packets, control the network flow, combine the traffic, split the traffic, load-sharing, hack the packets you can do via TrafficSqueezer. Hence it is not always a WAN Optimization solution.

TrafficSqueezer Configuration and Documentation

Along with the information in this documentation, you can also refer the above video specific to each TrafficSqueezer configuration scenario. In future more new videos to be uploaded and the same will be linked here for reference.

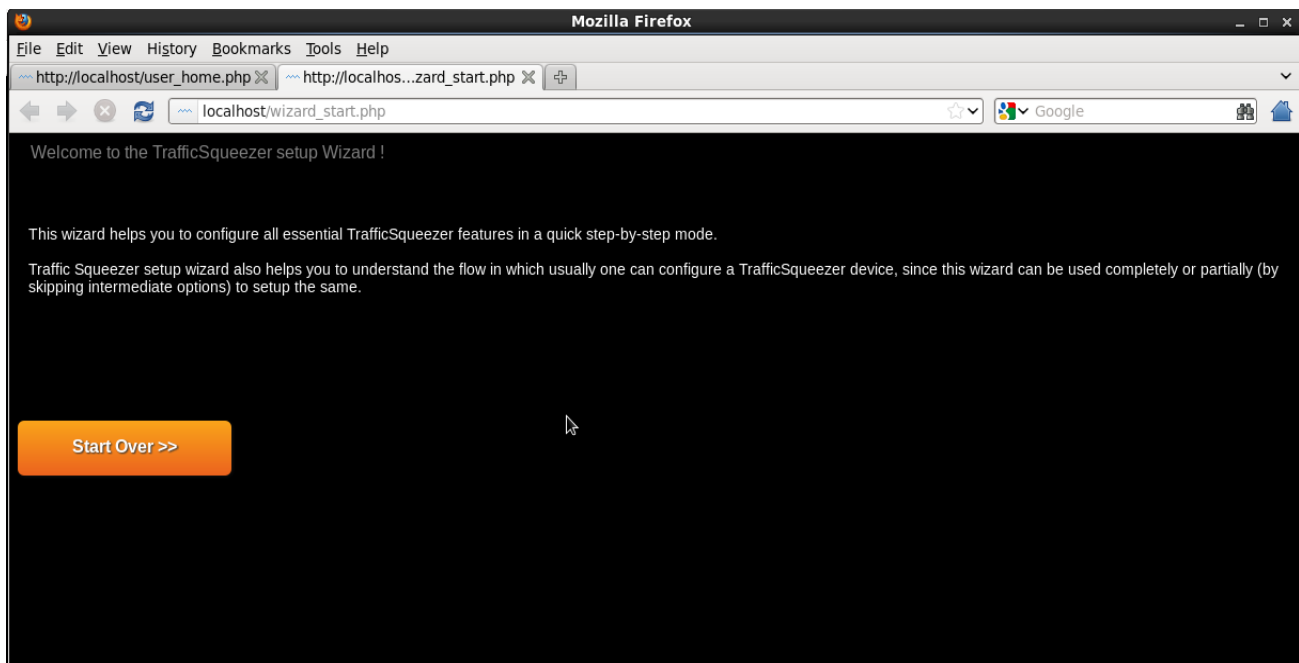
Tool-tips: Aquarium Traffic Squeezer configuration and settings modules contains tool-tips for each specific configuration option. Hence you can refer the same on-demand to know more about the same.

For example below is a screen-shot of Aquarium Tool-tip:



Wizard(s): Aquarium TrafficSqueezer wizard contains the important Traffic Squeezer user-friendly interactive configuration wizard assistance. With this users can usually configure and achieve up to 70-100% Traffic Squeezer device configuration objectives.

For example below is a screen-shot of Aquarium TrafficSqueezer wizard first page:



To know about combination of Wizards and settings components, you can refer the real-time dynamic videos.

TrafficSqueezer feature specific Documentation:

Since now you have feature specific tool-tips, you can refer the tool-tips basically to understand TrafficSqueezer feature specific Documentation. Also the wizard contains documentation in each step to further assist.

IMPORTANT NOTE – READ-ONLY Variables/Settings:

Users need to understand that each feature may contain one to many settings/variables. Now in order to increase the TrafficSqueezer run-time performance and also to reduce and refine the flow of the code and run-time logic, some of the variables are made as read-only. This

greatly reduces the time when users want to quickly setup a working TrafficSqueezer deployment. Due to the same, now instead of fiddling with cumbersome settings, users will have already a partially configured setup which is highly fine-tuned for great optimal performance.

TrafficSqueezer Run-Time performance, Packet Delivery and Latency

It is one of the core focus that we get maximum amount of possible WAN Optimization with maximum amount of packet delivery rate i.e effective throughput Mbps, and Gbps performance out of TrafficSqueezer. It is observed with thousands of trials on various systems that TrafficSqueezer needs effectively raw CPU processing power. I benchmark any PC/Server first via the total time taken to compile the Linux Kernel with full CPU load (i.e 100%). In other words a sheer MIPS is the best way to benchmark in an application like this. Since we are not using these systems as home PC or gaming PCs or servers or workstations, most of the CPU benchmarks done commercially is not much applicable for TrafficSqueezer kind of requirement.

Hence to achieve the same here are some of the steps undertaken:

- Introduction of Read-Only Variables.
 - Reduces the complications, less code, less latency per-packet.
 - Less run-time user settings. No cumbersome settings.

TrafficSqueezer (custom Linux Kernel) Installation

The download you get is a pack of TrafficSqueezer Kernel core a pre-patched Linux Kernel and packed with Aquarium GUI.

In other words it is just a virgin vanilla Linux Kernel (downloaded from www.kernel.org) patched with Open-Source TrafficSqueezer Kernel core source.

To compile the Kernel, you can follow the steps as already highly detailed and published in the TrafficSqueezer online documentation. Here are the URLs for reference:

Page1: http://www.trafficsqueezer.org/d_kernel_compilation.php

Page2: http://www.trafficsqueezer.org/d_kernel_compilation2.php

Page3: http://www.trafficsqueezer.org/d_kernel_compilation3.php

Page4: http://www.trafficsqueezer.org/d_kernel_compilation4.php

If you have issues with respect to hardware compatibility, device driver issues, you may need to patch it, and so build your custom kernel further.

Here are some frequent issues you face while using a custom kernel (includes even kernel downloaded from kernel.org):

- Kernel crash – due to missing proper device drivers, CPU incompatibilities and other Kernel module settings.
- Kernel crash – due to hardware incompatibilities.
- Failing to boot – due to non-enabling of specific partition and or file-system kernel modules.
- Failing to boot – due to boot loader issues.
- Failing to detect NIC Cards and other hardware – enable the specific Kernel driver modules before compilation of the same.
- Kernel crash/inconsistencies – perhaps some bug in some parts of the kernel code of that release, may not be completely related to TrafficSqueezer. Please remember not all the kernels are stable kernels. Kindly visit www.kernel.org for more details about stable, vs. experimental and intermediate kernel releases.
- Virtualization platforms (including VMWare): Sometimes these platforms may fail to more recent versions of kernel. That certainly includes TrafficSqueezer pre-patched kernel you get. Hence I strictly recommend either you first pre-customize according to the specifications of these Virtualized platforms. You may notice booting issues,

as well as crashes if it is not completely compatible. This is not a full-fledged kernel which is prepacked with Ubuntu/Fedora/CentOS distributions. It is a basic kernel.org kernel with TrafficSqueezer code in it.

Hence the above points may give you some idea about what you need to do and/or understand when there is an issue. This kernel which you are using is a general purpose kernel unlike a kernel pre-packed in Ubuntu/Fedora/CentOS, etc. distributions. The kernel in any Linux distribution is a highly tested, and pre-patched kernel.

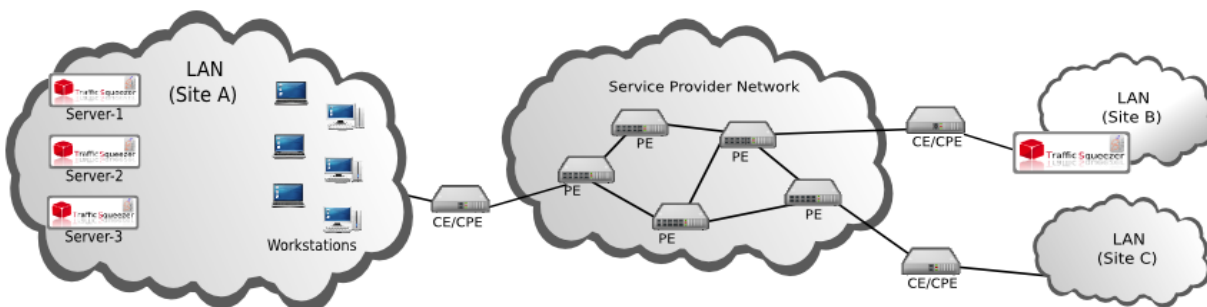
Hence to the extent of the knowledge required and the complexity, I suggest if users find it hard to get a working custom kernel, it is suggested to get some expert opinion and support.

Note: TrafficSqueezer does not support yet x86-64 bit based platforms. So use only 32-bit generic x86 platform as your test or deployment OS platform.

TrafficSqueezer Deployment Samples

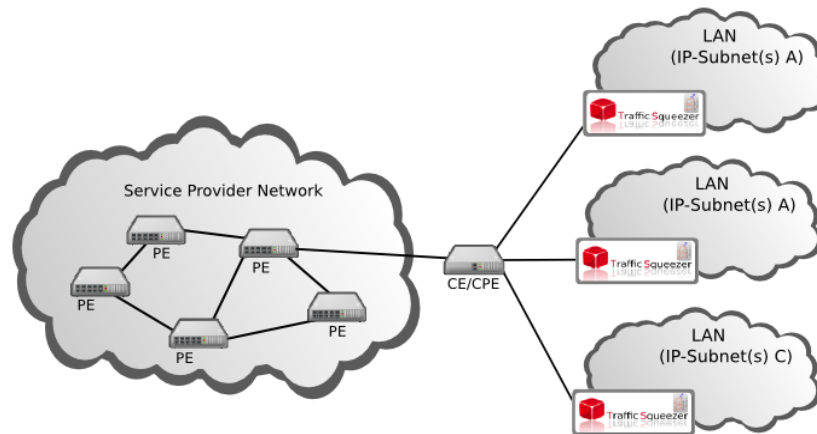
Traffic Squeezer – Server/Device Optimization

Traffic Squeezer can be installed in any server or device/machine, and can be made to send optimized packets to select remote IP subnets. This eliminates completely the use of one dedicated Traffic Squeezer router/bridge configured as a gateway. Traffic Squeezer features can be selectively enabled on any dedicated systems PCs or even servers (like DB server, Web servers, File servers, Email servers and so on). When this mode of optimization is enabled, the system by itself too gets capable of sending and accepting Traffic Squeezer optimized packets, besides which it may also work as a traditional routing device.



Load Sharing – Divide and Conquer

Assume there is a large enterprise setup, in order to share the packet processing load as well to achieve **Divide and Conquer** strategies, TS can be deployed as shown below. Since there are multiple *inexpensive* Traffic Squeezer devices, the single point of failure scenario is completely avoided and yet at the same time achieve load sharing advantages !



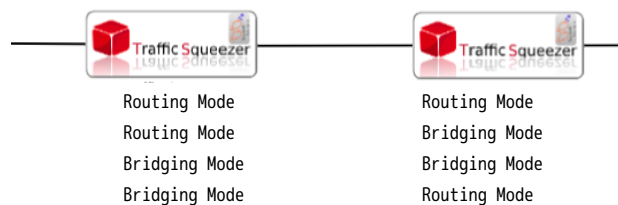
Load Sharing – Pipeline Mode

You can use multiple Traffic Squeezer machines in this configuration as shown below and enable various optimization features selectively in each Traffic Squeezer device so that the processing load can be shared across the machines. This strategy might help in a high-speed link. Say for example if one machine does compression, and so on, the other machine can perform packet templating, coalescing alone. This way in-expensive multiple Traffic Squeezer machines can be deployed. For even more critical links even more than two machines can be used or with a combination of other load sharing strategies with this.



Usually expensive network appliance devices covers all these features into a highly expensive dedicated hardware. But the obvious dis-advantage is that sometimes its tough to upgrade, as well sometimes it might introduce some limitations time to time with the upcoming infrastructure needs/requirements. This is the main reason where Traffic Squeezer although being a complete software solution can be used which always works in a generic computational platform. It is easy to upgrade the Traffic Squeezer software, as well easy to upgrade the hardware just in case.

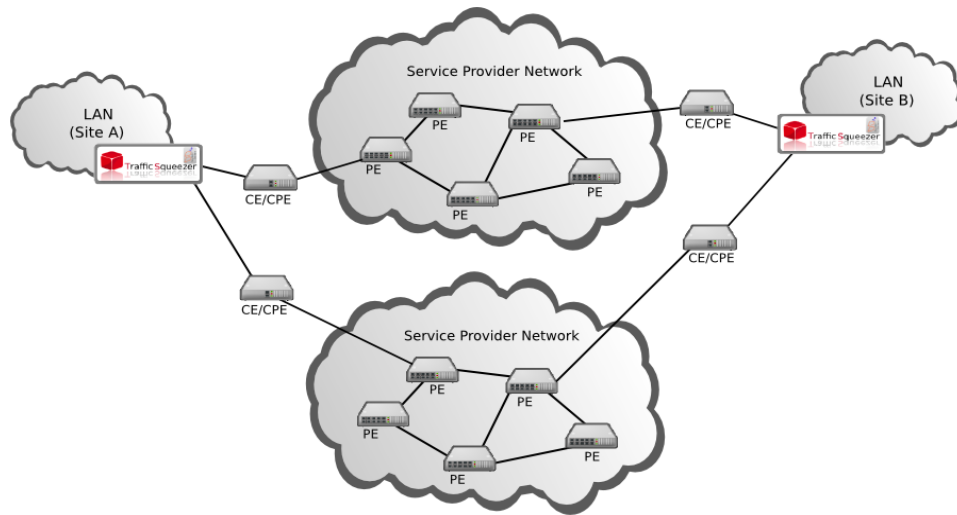
And not just that, even you can configure these systems one in Routing mode and other in Bridging mode. This way both these machine can be as shown below:



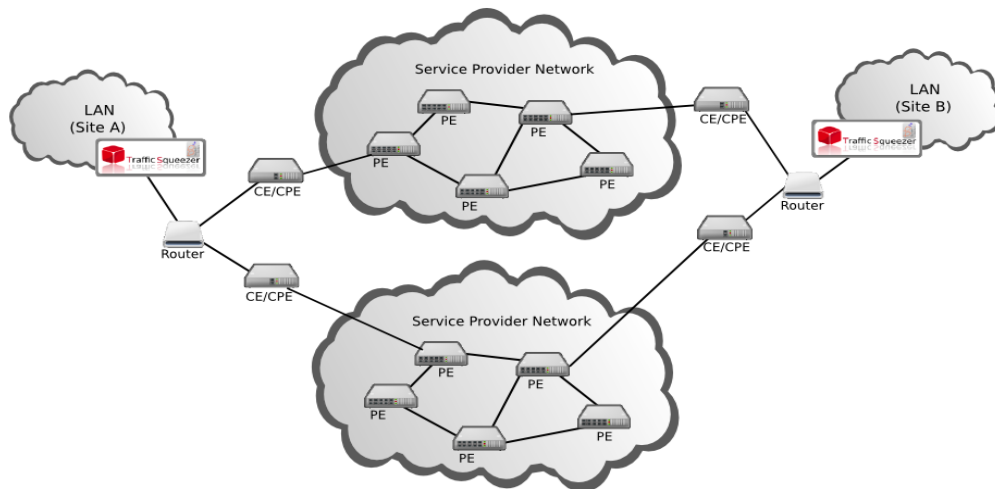
These creative configurations are purely situational as one can understand, Bridging mode usually reduces significant latency and packet processing, as well makes the presence of Traffic Squeezer even more transparent with respect to the deployed infrastructure.

Multi-vendor Connectivity

In case if the organization gets data connectivity via multiple vendors, (for the obvious reasons like load sharing and a backup secondary link and so on), TS can be deployed in these scenarios as shown below.

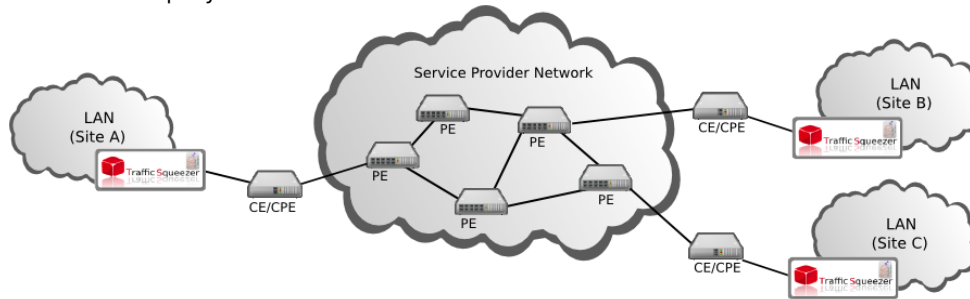


Similar deployment via an intermediate router.



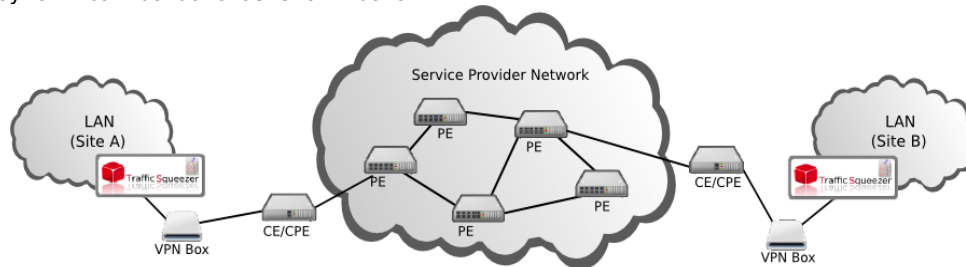
Multi-site Connectivity

In case if the organization have multiple sites (or multiple geographical branches), then in that case Traffic Squeezer can be deployed as shown below.



Connectivity via third-party VPN Device

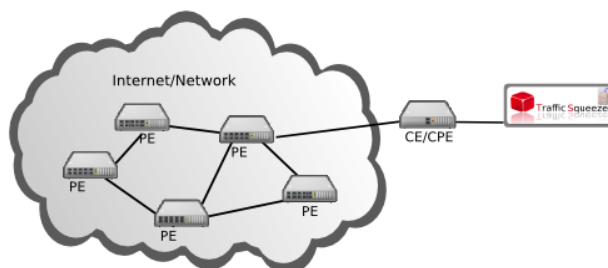
In case if there is a dedicated third-party VPN Device is already deployed, then in that case a Traffic Squeezer deployment can be done as shown below.



Warning: Never deploy TrafficSqueezer after a VPN device, it is obvious the traffic might get already encrypted in the VPN machine, this leaves not much room for TrafficSqueezer to perform any packet-wise optimization!

Simulation of TrafficSqueezer with 1 stand-alone machine

Simulation of TrafficSqueezer in server/device (local-device) mode. This way any stand-alone machine connected to Internet or any network can be simulated as if it does Traffic Squeezer configured optimization techniques. This simulates one direction of Traffic by optimizing it, and back un-optimizing (reverse optimization) it. Before the packets leaves the TrafficSqueezer device, gets compressed and optimized, and again gets un-compressed and un-optimized and sent out, while doing so, the statistics are captured, which gives an idea of the performance, and percentage of data optimization.



It is quite very easy to setup a single system simulation mode. Kindly refer this video which is a live-demonstration of the same.

Demonstration of Traffic Squeezer - Single system simulation mode

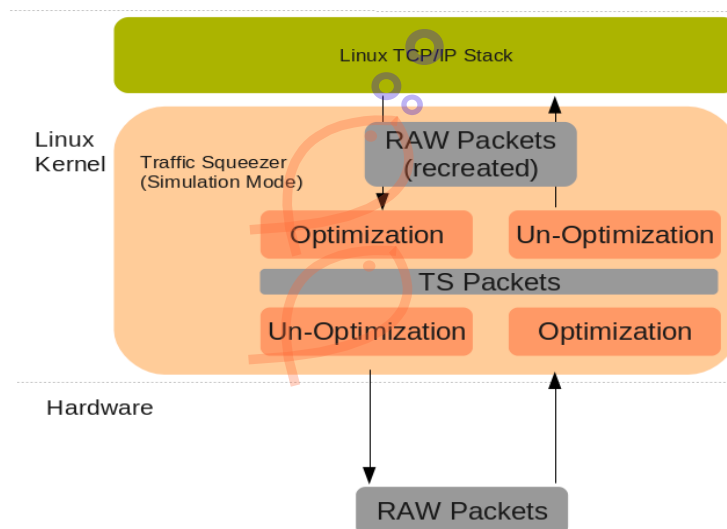
URL: <http://www.youtube.com/watch?v=dT9hXYtseA>

How and why Simulation mode is useful ?

- Saves your time.
- It is extremely easy to setup and test the Traffic Squeezer stack (except Coalescing feature, Coalescing feature will not work in simulation mode).
- #ping 127.0.0.1 (i.e you are testing completely your TCP/IP network stack right?) same way Traffic Squeezer simulation mode, you are testing 70-90% of Traffic Squeezer stack already.
- Besides, simulation mode helps you mainly to LEARN and UNDERSTAND Traffic Squeezer too.

Traffic Squeezer Optimization mode in simulation is extremely powerful. This does real real Traffic Squeezer traffic optimization and again does traffic un-optimization. This makes it compatible with the entire existing non-Traffic Squeezer infrastructure. The simulation is also tests 90% of the Traffic Squeezer Kernel Network stack capabilities and the results and benchmarking derived out of it is accurate up to 99% of that of the Traffic Squeezer device configured without simulation in a data real optimization setup.

Here is the reason why and how it works:



During simulation, TrafficSqueezer converts raw incoming and outgoing packets into TrafficSqueezer optimized packets, which again converted/recreated into raw packets. This way TrafficSqueezer simulates exactly TrafficSqueezer optimization in a non Traffic Squeezer network/infrastructure.

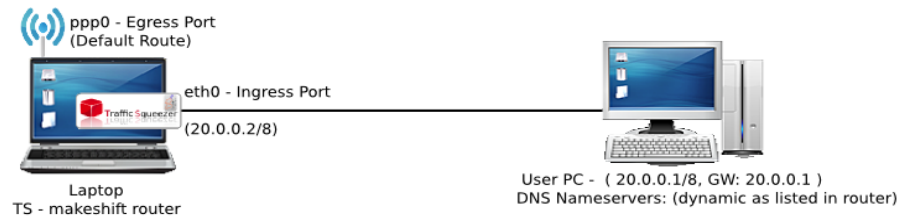
TrafficSqueezer Miscellaneous features

TrafficSqueezer – Makeshift Router

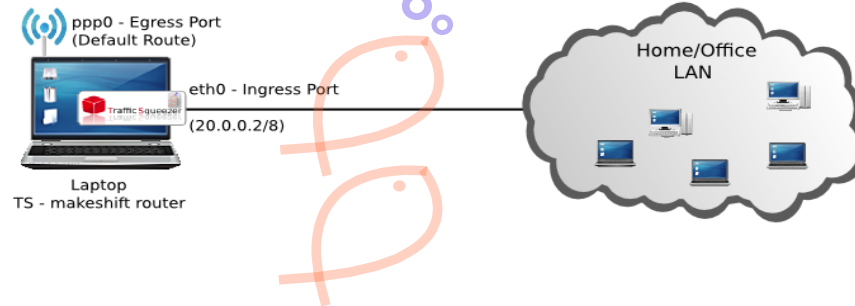
A quick make shift router can be quickly configured via TrafficSqueezer. Assume you have a old out-dated Laptop/PC, you can configure as a quick makeshift router, or as a backup router, or a stand-by router in case on an event of primary link service outage and so on. A makeshift router not only helps a home user, but can help even micro and mini business establishments.

Assume you have an out-dated laptop installed with Linux distribution. Now assume you have some form of internet CDMA/3G Data card, the same can be quickly used as a small office or a home router as shown below:

Sharing the link with one or couple of more systems via an outdated laptop:



Sharing the link with a small Home or Office (or SOHO) LAN:







WARNING: Please note that during a configuration like this, DO NOT enable TrafficSqueezer data optimization features which can alter the packet contents namely compression, coalescing, http-optimization, traffic templating and so on.

However, on a brighter side, users can use TrafficSqueezer other optimization techniques such as QoS service, Firewall service, http-caching, and so on, this way users can control the internet sharing and offer smooth data connectivity via link sharing like any other advance managed router or network appliance device !

TrafficSqueezer Statistics - Interpretation

In this section I am going to discuss the way you need to understand and interpret the TrafficSqueezer run-time statistics displayed in the Aquarium GUI.

Here is the screenshot of TrafficSqueezer Statistics:

Overall Stats [Compression + Packet Templating + HTTP-Optimization]: 					
INGRESS: rx 4145679 [3.95MB], tx 4040211 [3.85MB] - (Saved 105468 [103KB] 2.54%)					
EGRESS: rx 1949983543 [1.82GB], tx 12636588491 [11.77GB] - (Saved 10886804948 [9.95GB] 84.57%)					
Compression Stats:					
INGRESS: rx 2354284 [2.25MB], tx 2239824 [2.14MB] - (Saved 114480 [111.78KB] 4.86%)					
EGRESS: rx 157961159 [150.64MB], tx 10877436852 [10.13GB] - (Saved 10719475493 [9.98GB] 98.55%)					
HTTP Optimization Stats:					
INGRESS: rx 3812 [3.53KB], tx 2500 [2.44KB] - (Saved 1112 [1.09KB] 30.79%)					
EGRESS: rx 5313907 [5.07MB], tx 5370739 [5.12MB] - (Saved 56832 [55.5KB] 1.06%)					
Template Dictionary Stats:					
INGRESS: rx 0 [0], tx 0 [0] - (Saved 0 [0] 0%)					
EGRESS: rx 0 [0], tx 0 [0] - (Saved 0 [0] 0%)					
Coalescing Stats: 					
INGRESS: rx 0 [0], tx 0 [0] - (Saved 0 [0] 0%)					
EGRESS: rx 8853481 [8.44MB], tx 8853481 [8.44MB] - (Saved 0 [0] 0%)					
Filter DNS Stats:					
INGRESS: 0 filtered packets					
EGRESS: 0 filtered packets					
Packet sizes Stats: 					
INGRESS			EGRESS		
size	in	out	size	in	out
<64	2489	2489	<64	7821547	17791
64_127	9774	73055	64_127	16822	16822
128_255	2272	3042	128_255	9184	79610
256_511	1775	1104	256_511	18281	123558
512_1023	431	332	512_1023	32273	284582
>1024	730	730	>1024	959688	8270151
IP Protocol Stats: 					
INGRESS:			EGRESS:		
TCP :	20995 pkts		TCP :	8848983 pkts	
UDP :	3324 pkts		UDP :	64 pkts	
ICMP :	0 pkts		ICMP :	0 pkts	
SCIP :	0 pkts		SCIP :	0 pkts	

First you need to understand is that there are various categories of statistics. Some of them are related to TrafficSqueezer statistics, and some are actually a break-up of generic statistics. I.e:

- Overall Statistics
 - Compression Statistics
 - HTTP Optimization Statistics
 - Templating Statistics
- Coalescing Statistics
- Filter DNS Statistics
- Packet Sizes Statistics
- IP Protocol Statistics

INGRESS:

Ingress denotes the flow of traffic from LAN to WAN. Hence it is actually a flow from Ingress to Egress, which in short represented as “INGRESS”. Hence you should always interpret as Ingress->Egress traffic flow.

EGRESS:

Egress denotes the flow of traffic from WAN to LAN. Hence it is actually a flow from Egress to Ingress, which in short represented as “EGRESS”. Hence you should always interpret as Egress->Ingress traffic flow.

Direct TrafficSqueezer optimization statistics (optimization means I refer here as some data optimization via TrafficSqueezer), are represented via “Overall Optimization Statistics” and via “Coalescing Statistics”. Where-as indirect optimization statistics are represented via “Packet Sizes Statistics” and “IP Protocol Statistics”. If TrafficSqueezer is disabled (i.e no optimization is enabled), then Packet Sizes Statistics and IP Protocol Statistics will be quite representative, and

will clearly show whatever received in Ingress port, is forwarded without any change/optimization to Egress port. And similarly whatever received in Egress port, is forwarded to the Ingress port.

TrafficSqueezer - Overall Optimization Statistics:

This is a measure which deals with "DATA or PACKET SIZE REDUCTION". Hence it is classified as such. This is the measure of overall combined optimization i.e via Compression, Templating, and HTTP Optimization. Hence I call it as Overall Optimization Statistics. Please note Coalescing is a packet COUNT reduction technique, (and not packet-size reduction technique), hence it is not combined with this measure.

TrafficSqueezer - Coalescing Statistics:

This is a measure which deals with "PACKET COUNT REDUCTION". Since it has nothing to do with Packet-size reduction, I do not classify it with optimization statistics. Instead strictly Coalescing is an independent technique, and hence its Statistics is an independent measure.

TrafficSqueezer - Packet Sizes Statistics:

This is the most wonderful statistics than all. The reason is, once TrafficSqueezer is enabled, we can see "TRANSITION" of large packets getting converted into smaller packets in INGRESS flow. Whereas small packets are getting "RECREATED" into large packets in EGRESS flow. Hence it is a direct proof of TrafficSqueezer Optimization (Data Reduction) strategy. On the other hand, if there are any packets which are "MISSING" to get forwarded, then it is a direct proof of operation of TrafficSqueezer Coalescing Strategy. Hence packet Sizes Statistics is a very important visible tool to quantify its effectiveness.

TrafficSqueezer - IP Protocol Statistics:

As the name suggests it is a simple IP Protocol-wise Statistics. There is no fun in this statistics in regular context. But however, many users do not know that "TrafficSqueezer is a packet engineering tool besides a WAN Optimization solution", in this context if required packets can be hijacked/hacked, and can be converted into TCP->UDP, and UDP->TCP formats. It may be very useful in various contexts in a real world scenario, to bypass intermediate so-called intelligent networking devices, IDS, IPS systems. If so it is done, then IP Protocol Statistics is the only window, we can see this transition in real-time.
