Developer Guide

Laguna 1.0

CONCURRENT

Copyright Notice

Copyright 2015 Concurrent Computer Corporation

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

Refer to the License for the specific language governing permissions and limitations under the License.

License

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions.

Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Contact Us

Concurrent Computer Corporation

4375 River Green Parkway, Suite 100
Duluth, GA 30096 USA

GitHub: https://github.com/concurrentlabs/laguna
Email: concurrentlabs@concurrent.com

Contents

Chapter 1

1 OVERVIEW

Laguna is a transparent caching control plane providing IP/MPLS traffic monitoring & analysis, cache definition management & traffic matching.

Laguna monitors video traffic streams in an “out-of-band” manner for unencrypted packets. It then selectively re-directs video-related client requests to edge caches, based on configurable cache definition profiles and policies.

1.1 RELATED DOCUMENTS

- Libpcap: http://www.tcpdump.org/
- Pfring: http://www.ntop.org/products/pf_ring/
- Liblfds: http://www.liblfds.org/
- C Yaml Parser: http://pyyaml.org/wiki/LibYAML
- Zlog: http://hardysimpson.github.io/zlog/
- Nginx: https://www.nginx.com/resources/wiki/

1.2 DEFINITION OF TERMS

Acronym or Term	Definition
TCS	Transparent Caching System
CP	Control Plane
MTU	Maximum Transmission Unit
TBD	To be defined

Chapter 2

2 THEORY OF OPERATION

2.1 DESCRIPTION OF FUNCTION

The Concurrent transparent caching system is targeted with transparently caching Internet video content that is hosted on web sites external to the operator's network. There need be no business agreement between the operator and the web site in order to cache the content on the edge cached ("TCS engines") within the operator's network. The caching happens transparently, without requiring changes on the web site or subscriber's equipment.

The Laguna transparent caching system is deployed out of band.

2.2 NETWORK INTEGRATION

2.2.1 OUT OF BAND

The Laguna control plane of the transparent caching system (TCS) receives IP traffic via a tap in the network. This tap is an interface on a network switch, and the "tapping" can be done anywhere in the network where Internet traffic from an operator's subscribers can be accessed. The tap can be either an optical tap or port mirroring from a switch.

The purpose of the tap is to relay this Internet traffic to the Laguna component of TCS. This relay occurs in parallel to the outbound transfer of the traffic out to the Internet. Thus, a subscriber's Internet traffic travels simultaneously both to the Control Plane as well as to the Internet site requested by the subscriber.

Laguna monitors the traffic and parses the data to determine if the content is potentially cacheable based on cache headers, white lists, content length, content popularity or other criteria. If the content matches the potentially cacheable criteria, Laguna hands off the details to the Traffic Router. This in turn instructs an edge cache to respond to the client with the content by redirecting the client with a 302-Redirect response and terminates the client-Origin connection.

The client typically remembers the redirection so subsequent requests for data go direct to the edge cache.

Client authentication request exchanges are not handled by the Laguna control plane so once the first data request is seen, the business logic will already have been validated, requiring no additional checks from the TCS.

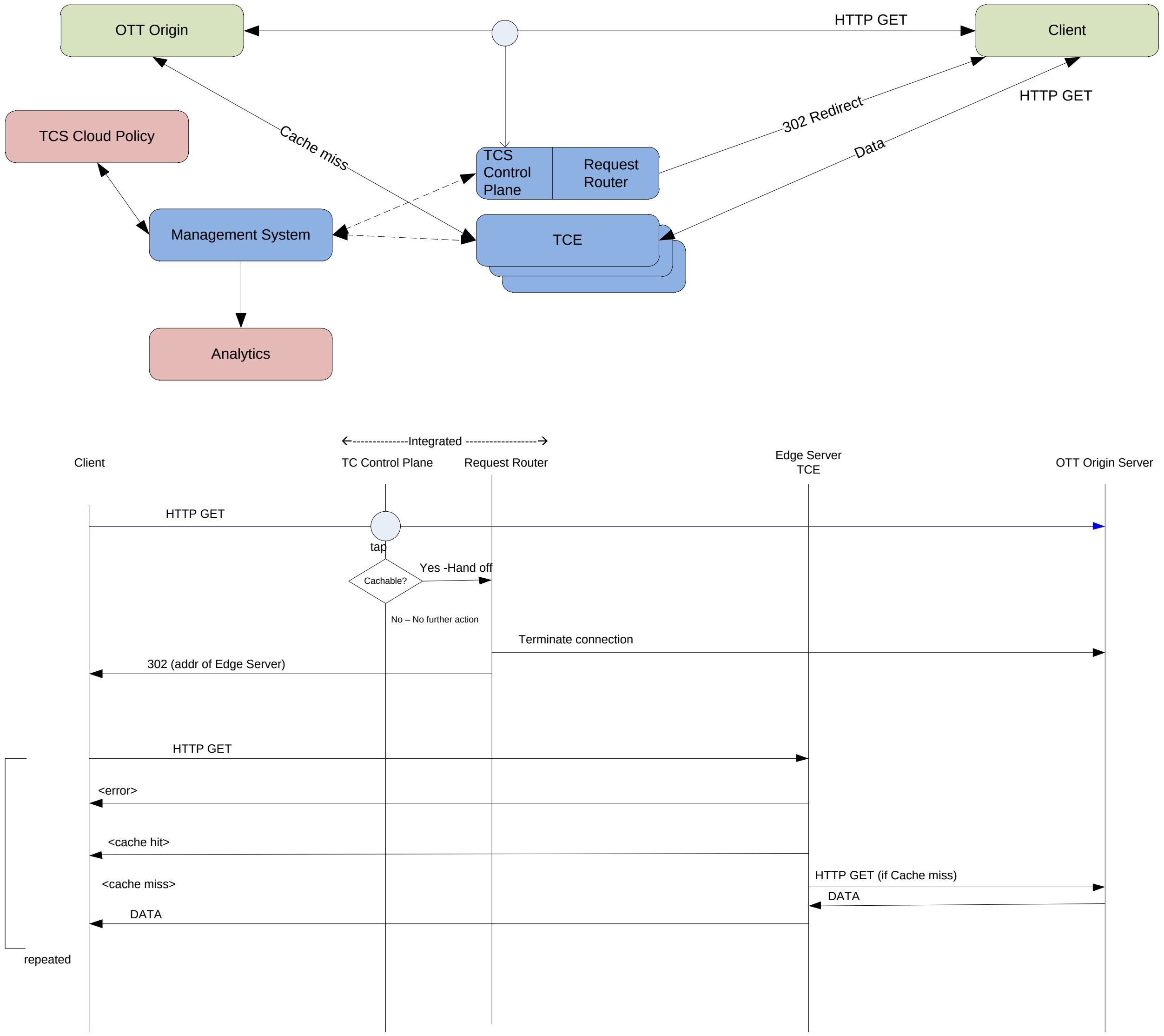


Fig 1 Simplified Data Flow – Out of Band

Chapter 3

3 Application Design

3.1 Overview

3.1.1 Architectural Tenets

- Modularized components for ease of maintenance and scalability
- Lock free queues communications with minimal mutex locking operations (component initializations and health checking)
- Multi-threaded shared memory with three part components: main, background and add-ons with subcomponents

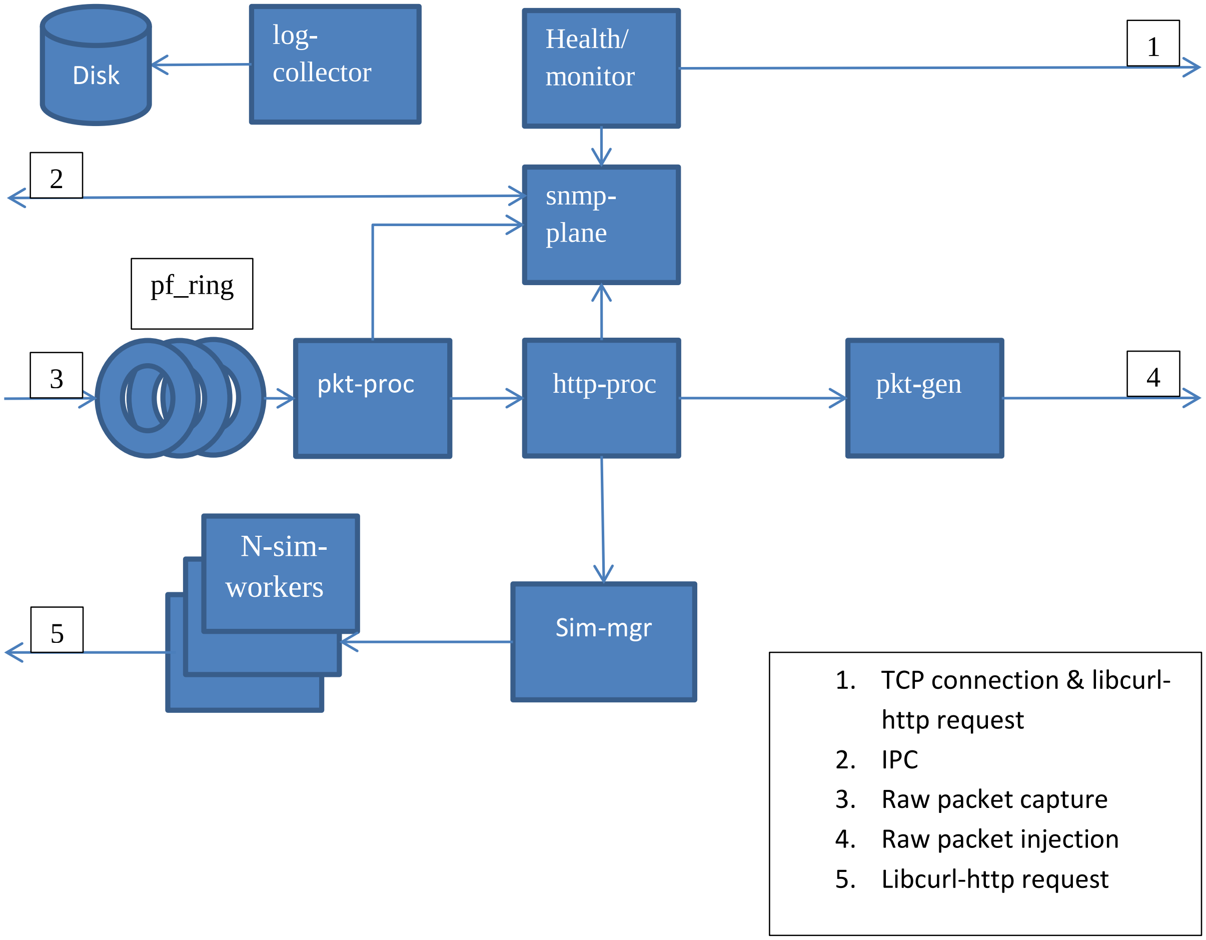
3.1.2 Components

1. Main:
    a. Packet processing (pkt-proc)
        i. Thread initialization and runtime configuration loading
        ii. Raw packet read from the wire (pf_ring)
        iii. Raw packet parsing (pf_ring)
        iv. L2-L4 Raw packet filtering (pf_ring)
        v. Minimal HTTP URL payload filtering
        vi. Queue message logging of thread runtime info and stats
        vii. Queue message write of domain name table message to snmp-plane
        viii. Queue message write of packet descriptor message to http-proc
    b. Http processing (http-proc)
        i. Thread initialization and runtime configuration loading
        ii. Full HTTP payload parsing
        iii. Cache key construction
        iv. Request Router max bandwidth limit check
        v. TCP video stream session correlation
        vi. Queue message logging of thread runtime info and stats
        vii. Queue message read of packet descriptor message from pkt-proc
        viii. Queue message write of redirection table message to snmp-plane
    c. Packet generation/injection (pkt-gen)
        i. Thread initialization and runtime configuration loading
        ii. Raw packet injection construction
        iii. Raw packet HTTP 302 redirection injection
        iv. Raw packet TCP RST injection
        v. Queue message logging of redirected services

vi. Queue message logging of thread runtime info and stats

2. Add-ons:
   a. Simulation manager (sim-mgr)
   b. Thread initialization and runtime configuration loading
      i. Byte range calculation if exists within http request
      ii. Dynamic cache key session tracking
      iii. Load balancing of curl http requests to sim-worker threads
      iv. Queue message read of packet descriptor message from http-proc
   c. Simulation workers (sim-workers)
      i. Thread initialization and runtime configuration loading
      ii. Curl HEAD and GET request operations
      iii. Byte range calculation from HTTP OK or 206 response
      iv. Dynamic cache key calculation
      v. Queue message read of packet descriptor message from sim-mgr
3. Background:
   a. Health/Monitor
      i. Thread initialization
      ii. Periodic check on all components health and init status
      iii. Periodic health check on request router or edge
      iv. Queue message write of health status to snmp-plane
   b. Snmp-plane
      i. Thread initialization
      ii. IPC (Inter process communication) read and write messages from/to agentx through ZeroMQ
      iii. Queue message read of domain name table message from snmp-plane
      iv. Queue message read of redirection table message from snmp-plane
      v. Queue message read of health status reporting from health/monitor
   c. Log-collector
      i. Thread initialization
      ii. Disk write of all log and report messages
      iii. Queue message read of all log messages from all components on the system
      iv. Queue message read of simulation report messages from simulation workers and manager

The following diagram illustrates the interaction of these components:



* http-proc and pkt-gen are different components but currently combined as one thread.

** Log-collector queue communications with all components on the system are not shown.

3.2 PROCESS STRUCTURE

3.2.1 PACKET ROUTING

The Laguna control plane pulls unencrypted packets from the wire through the libpcap/pfring component, performs transport layer filtering, decodes the packet from Layer 2 all the way to layer 4, grabs all the information such as packet length (caplen/wirelen), ip address, MAC address, VLAN Tag, TCP sequence and acknowledgement number and HTTP payload. All IP fragmentations are handled by packet capture library while TCP fragments are handled by packet processing thread.

Packet information is then passed into HTTP processing where policy rules are applied to the HTTP GET request to see whether the video packet needs to be routed or not. Some information is extracted from the request based on config options and then passed to the client through a 302 redirect.

Information extracted includes video server address, cache key id, other cache key info, options and original URL signature.

Example URL:

```
HTTP/1.1 302 Found
Location:
http://10.75.25.112/ccur/video/ntflx/tcshost/108.175.40.97/tcskey/1221124515.
ismv/35234922-37313717/tcsopt/cache/tcsosig/1221124515.ismv/range/35234922-
37313717?c=us&n=7029&v=3&e=1394260506&t=QwF54XNfX-
cryLOImpRV56jIdds&d=silverlight&p=5.NHtgCDE9GnpNGScFmyng7uQL27nCYm4tinZl3fuf3
YM&random=260770715
```

Additional URL information is then prepended (bold text above) into the HTTP 302 request before the message is sent to the client; FIN requests are also sent to the origin server. Raw packets are then fabricated and injected into the network coming from client to the server or vice versa. The 302 URL request will point the client to the caching server proxy, which will perform reverse proxy request to the appropriate origin server.

The caching server will then pick all the prepended information and classify the request based on information given. The OK response will then be cached based on combination of cache key id + range + misc. info supplied within the URL GET request coming from client.

If the GET request entry is not found within the cache (cache miss), then the cache server will then contact the origin server using address given. It is also important to note that a policy permission file, such as crossdomain.xml will also be cached by the cache server to service client requests.

3.3 PROGRAM INTERFACES

This section defines the fabricated TCP RST being sent to the internet origin server and the HTTP redirection request interaction with third party client browsers and caching server or reverse proxy components.

3.3.1 TCP RST+ACK REQUEST:

- Client server IP address, port and MAC Address
- Origin server IP address, port and MAC Address
- TCP RST+ACK Flag set.
- Fabricated next TCP Sequence number
- Fabricated TCP ACK number
- Less than 1500 MTU, not fragmented TCP

3.3.2 HTTP REDIRECTION REQUEST:

- Client server IP address, port and MAC Address
- Origin server IP address, port and MAC Address
- HTTP payload of 302 Found with date, content type text and length zero

Example:

```
Content-Type: text/html; charset=UTF-8
Date: Tue, 04 Mar 2014 15:37:07 GMT
Content-Length: 0
```

- Less than 1500 MTU, not fragmented TCP
- HTTP (302/303) URL Format**:**

  http://<edge proxy hostname address>/ccur/<site type>/<site target>/tcshost/<video server host address>/tcskey/<cache key id >/tcsopt/<options>/tcsosig/<site type>/<site target>/<original URL signature>

  - <edge proxy hostname address>: Hostname of caching server
  - <site type>: User defined monitored site content type such as: video, osupdate
  - <site target>: User defined monitored site content name such as: ytb, ntflx, apple, microsoft, etc
  - <video server host address>: Video server origin address
  - <cache key id>: unique stream id followed by misc stream info for caching separated by "/", zero if not specified.
  - <options>: user defined caching options.
  - <original URL signature>: original URL signature from the client to the origin server

Example Http Payload:

```
HTTP/1.1 302 Found^M
Location: http://10.75.25.112/ccur/video/ytb/tcshost/r14---sn-
5uaeznlz.googlevideo.com/tcskey/92e084af28144bf8/135-1712128-
3424255/tcsopt/cache/tcsosig/videoplayback?clen=7448966&cpn=1v43P4Q_Tq5IY2SU&
dur=111.867&expire=1394259031&fexp=919120%2C911429%2C932280%2C943104%2C916626
%2C937417%2C913434%2C936910%2C936913%2C902907%2C934022&gir=yes&id=92e084af281
44bf8&ip=173.221.58.2&ipbits=0&itag=135&keepalive=yes&key=yt5&lmt=13900972889
70083&ms=au&mt=1394233812&mv=u&mws=yes&nh=IgpwcjAxLmF0bDAxKgkxMjcuMC4wLjE&ran
ge=1712128-
3424255&ratebypass=yes&signature=4622578DDAFAEAAA07424005DCADF6180F4C1F1F.AB1
DF9A5BC2AC32CA981553F36C9C31661B1F259&source=youtube&sparams=clen%2Cdur%2Cgir
%2Cid%2Cip%2Cipbits%2Citag%2Clmt%2Csource%2Cupn%2Cexpire&sver=3&upn=dXolCxUlw
KA
Content-Type: text/html; charset=UTF-8
Date: Fri, 07 Mar 2014 23:13:00 GMT
Content-Length: 0
```

3.4 CONSTRAINTS

- The system must be run with super user mode permissions due to the promiscuous nature of IP traffic monitoring.
- There must be a mirrored port (SPAN) or network tap interface connected to the Laguna control plane monitoring port for it to analyze traffic and apply interception policies.

Chapter 4

4 CONFIGURATION

4.1 CONFIGURATION

The Laguna control plane can be configured to process HTTP request message to do the following operations:

1. Packet monitoring, filtering and routing

   Performs deep packet inspection on unencrypted packets and route them to any specified targets based on specified URL (mandatory) and HTTP header hostname pattern (optional). During HTTP process of pattern matching, the configuration file pattern is processed top to bottom.

2. Cache key retrieval and creation

   Full cache key pattern is composed of cache id, range and other information. Cache key id is the video asset unique identifier; range is the video byte range request while other information is any additional information. Miscellaneous cache key is composed of range and other additional information. Cache key id and miscellaneous values can be retrieved from the control plane side or through mapping on the edge side through special */services/options* field from table1 below.

3. Video stream context session tracking

   Video session tracking is being done strictly through URL and HTTP header pattern matching and not IP address-port tracking. A session may have multiple contexts that can be grouped together as one context group per session if need be. Hash table of key-value pair is being used to maintain session contexts and grouping.

Based on the above fundamental operations, there are two types of configuration options, simple and advanced.

1. Simple: Simple configuration performs packet monitoring, filtering and routing, cache key retrieval and creation operation without the complexity of video stream context session tracking. In short, each HTTP request is independent to other future requests.
2. Advanced: All simple configuration features with the addition of video stream context session tracking. Session context tracking is being done by building relationships between HTTP requests. In short, each HTTP request is dependent to other future requests.

4.1.1 CONFIG.YAML

Control plane runtime and pattern matching reloadable configuration structures.

Tag Name	**Description**	**Unit**	**Limits**	**Optional**	**Relationships**
/version	Config version	String	32 bytes	No	None
/modeofoperation	Mode of operation: “active” or “monitor”	String	32 bytes	No	None
/ monitoringinterface	Multiple monitoring interface options separated by “;”. *Format: <intf-name>:<intf-direction>;….* *<intf-direction>:* rx	String	512 bytes	No	/pcap-filter

Tag Name	Description	Unit	Limits	Optional	Relationships
	tx example: “eth0:rx”;”eth1:tx” common usage: “eth0:rx”				
/pcap-filter	Multiple packet capture filter options separated by “;”. *Format: <intf-name>:<intf-filter>;….* *<intf-filter>:* port80-src port80-dst example: “eth0:port80-dst”;”eth1:port80-src” common usage: “eth0:port80-dst”	String	1024 bytes	No	/ monitoringinterface
/ outgoinginterface	Outgoing interface	String	64 bytes	No	None
/redirectaddress	Request Router or caching server address also serves as black list address.	String	1024 bytes	No	None
/bwsimulationmode	Bandwidth simulation mode. “true” for active or “false” for non-active simulation mode.	String	8 bytes	No	/bwsimulationworkers
/bwsimulationworkers	Number of simulation worker threads. By default 15 workers will be created if on active mode and only 1 on non-active simulation mode. This value is static and not a reloadable configuration.	String	16 bytes	Yes	/bwsimulationmode
/bwsimulationoutgoinginterface	Bandwidth simulation mode outgoing interface. It will try to pick the interface through routing table if not specified.	String	64 bytes	Yes	None
/ipblacklist	Black list of ip address separated by “,”.	String	2048	Yes	None
/services	List of interested monitored target content services	List	32 objects	No	
/services/type	Content type	String	64 bytes	No	None
/services/target	Content target name	String	256 bytes	No	None

Tag Name	Description	Unit	Limits	Optional	Relationships
/services/options	Edge cache server site content options. Value options strictly being passed to edge server. Available options: ▪ "nocache", Force edge caching server to do HTTP reverse proxy ▪ "cache", Force edge caching server to do HTTP reverse proxy and cache the video content ▪ "<service>-ckeymap-<container file type>-<checksum range>"	String	32 bytes	No	None
/services/hostsignature	Content HTTP header hostname signature pattern separated by ";"	String	256 bytes	Yes	None
/services/referersignature	Content HTTP header referrer signature pattern separated by ";". Overloads content cache key Id value.	String	256 bytes	Yes	/services/url/cachekeyid
/services/httprangefield	Content HTTP header range field name. Overloads content cachekey range value.	String	256 bytes	Yes	/services/url/cachekeyrange
/services/httpsessionfield	Content HTTP header session field name. Media stream contexts are grouped by session id if specified.	String	256 bytes	Yes	/services/url/contextid
/services/url	List of content GET request URL signature	List	64 objects	No	None
/services/url/signature	Content GET request URL signature	String	1024 bytes	No	None
/services/url/maxmatchsize	Maximum URL GET request signature match size length	Unsigned int	128 bytes	No	None
/services/url/contextid	Multiple content context Id URL relationships to none or list of sessions signature patterns for HTTP GET separated by ";". Context is grouped by session id if specified.	String	1024 bytes	No	/services/session/contextid, /services/httpsessionfield

Tag Name	Description	Unit	Limits	Optional	Relationships
/services/url/cachekeyid	Multiple content cache key Id URL signature patterns for HTTP GET separated by “;”. Being Overloaded by HTTP header referrer field value.	String	1024 bytes	No	/services/referer signature
/services/url/cachekeyrange	Content cache key range URL signature pattern for HTTP GET. Being Overloaded by HTTP header range field value. This value should be in http range format or leave it empty if none exists then “0-“ will be set for the range.	String	1024 bytes	Yes	/services/httprangefield
/services/url/cachekeymisc	Multiple content cache key miscellaneous URL signature patterns for HTTP GET separated by “;”	String	1024 bytes	Yes	None
/services/session	List of session GET request signature	List	64 objects	Yes	None
/services/session/signature	Session GET request signature	String	1024 bytes	No	None
/services/session/maxmatchsize	Maximum session GET request signature match size length	Unsigned int	128 bytes	No	None
/services/session/cachekeyid	Multiple session cache key Id signature patterns for HTTP GET separated by “;”.	String	1024 bytes	No	None
/services/session/contextid	Multiple session context Id relationships to list of urls signature patterns for HTTP GET separated by “;”	String	1024 bytes	No	/services/url/contextid
/services/session /cachekeyrange	Content cache key range session signature pattern for HTTP GET. Being Overloaded by HTTP header range field value.	String	1024 bytes	Yes	/services/httprangefield
/services/session /cachekeymisc	Multiple content cache key miscellaneous session signature patterns for HTTP GET separated by “;”	String	1024 bytes	Yes	None

4.1.2 Simple configuration

Example 1:

```
   - type: 'video'
      target: 'ytb'
      options: 'cache'
      hostsignature: '"(googlevideo.com)+";"(youtube.com)+"'
      referersignature: '"/watch\?v=([^\&]+)";"/embed/([^\&]+)"'
      httprangefield: 'Range'
      url:
            - signature: '(/videoplayback/id/)'
              maxmatchsize: 20
              cachekeyid: '/videoplayback/id/([^/]+)'
              cachekeyrange: '/go[a-z]p/([^/]+)/begin/([^/]+)'
              cachekeymisc: '"/itag/([^\/\?]+)";"/file/([^\/\?]+)"'
            - signature: '(/videoplayback).*?([\&\?]id=([a-z0-9]{2}))'
              maxmatchsize: 600
              cachekeyid: '[\&\?]id=([^\&]+)'
              cachekeyrange: '[\&\?]range=([^\&]+)'
                    cachekeymisc: '[\&\?]itag=([^\&]+)'
```

Let’s break down above configuration into easier terms, the content *type* is “video” of “ytb” *target* site with the *option* for the edge server to “cache” the content. *hostsignature* specifies control plane to narrow the filtering of hostname scope to certain hostnames sites after a match occurs on one of the two URL *signature* patterns with *maxmatchsize* URL limit of 20 and 600 bytes.

If *maxmatchsize* is bigger than the URL length, the full length of the URL will be used instead. The *url* field describes n-number of ways on how to perform packet filtering, routing, cache key retrieval and creation.

cachekeyid will be unique video asset identifier that has specific *cachekeyrange* followed by additional information such as codec type or any other miscellaneous information of *cachekeymisc* URL pattern.

About cache key id and range overloading:

Some request provides cache key through HTTP header referrer signature **referersignature**.

If specified, this information will overload URL **cachekeyid** value. The same for the HTTP range field. HTTP **httprangefield** can be used to overload URL **cachekeyrange**. Since both cache key range and miscellaneous are optional fields, both values will be nulled if not specified.

Example 2:

```
    - type: 'video'
      target: 'ntflx'
      options: 'ntflx-ckeymap-mp4-0_256'
      url:
            - signature: '/range/.*?([\?]o=AQ).*?v=.*?t='
              maxmatchsize: 250
              cachekeyid: '[\&\?]o=([^\&]+)'
            - signature: '//range/.*?([\?]o=AQ).*?v=.*?t='
```

```
              maxmatchsize: 250
              cachekeyid: '[\&\?]o=([^\&]+)'
            - signature: '\/\?o=AQ.*?v=.*?t='
              maxmatchsize: 250
                   cachekeyid: '[\&\?]o=([^\&]+)'
```

While everything else is the same as previous configuration, the control plane uses option field (bold letter) to pass special instructions to edge server on how to retrieve cache key id value. From table 1 *"/services/options",* the option service is Netflix ("ntflx"), container type MP4 ("mp4") with checksum range of 0 to 256 ("0_256").

Above configuration option is requesting edge server to perform dynamic cache key id mapping of *cachekeyid* to static cache key id using MP4 header container file from HTTP response with the range of 0 to 256. Since cache key range is not being specified, this value will have to be pulled by edge server.

4.1.3 ADVANCED CONFIGURATIONS:

Example 1:

```
    - type: 'video'
      target: 'ytb'
      options: 'cache'
      hostsignature: '"(googlevideo.com)+";"(youtube.com)+"'
      referersignature: '"/watch\?v=([^\&]+)";"/embed/([^\&]+)"'
      httprangefield: 'Range'
      httpsessionfield: 'X-Playback-Session-Id'
      session:
            - signature: '(/stream_204\?)'
              maxmatchsize: 16
              contextid: '[\&\?]cpn=([^\&]+)'
              cachekeyid: '[\&\?]docid=([^\&]+)'
            - signature: '(/ptracking\?)'
              maxmatchsize: 16
              contextid: '[\&\?]cpn=([^\&]+)'
              cachekeyid: '[\&\?]video_id=([^\&]+)'
      url:
            - signature: '(/videoplayback/id/)'
              maxmatchsize: 20
              cachekeyid: '/videoplayback/id/([^/]+)'
              cachekeyrange: '/go[a-z]p/([^/]+)/begin/([^/]+)'
              cachekeymisc: '"/itag/([^\/\?]+)";"/file/([^\/\?]+)"'
            - signature: '(/videoplayback).*?([\&\?]id=([a-z0-9]{2}))'
              maxmatchsize: 600
              cachekeyid: '[\&\?]id=([^\&]+)'
              cachekeyrange: '[\&\?]range=([^\&]+)'
                    cachekeymisc: '[\&\?]itag=([^\&]+)'
            - signature: '(/videoplayback\?).*?(c=[Ww][Ee][Bb])'
```

```
              maxmatchsize: 800
              contextid: '[\&\?]cpn=([^\&]+)'
              cachekeyrange: '[\&\?]range=([^\&]+)'
              cachekeymisc: '[\&\?]itag=([^\&]+)'
```

The first two url configuration entries are explained on the simple configuration option and everything else is the same as simple configuration except the relationships between httpsessionfield and contextid (bold values).

- *httpsessionfield* groups n-numbers video session contexts into a group of context with the same string value.
- *contextid* is a context pattern for control plane to track and build relationships between multiple HTTP requests for one video session.

In this case, URL contextid pattern is specified as youtube “cpn” field value identifier, which is linked to another request contextid pattern per video session. session/cachekeyid signature pattern within session field is being used to capture the video session cache key while the range and miscellaneous values are being captured through url/cachekeyrange and url/cachekeymisc config options

In short, the video session context is being tracked through “cpn” value and grouped together by “X-Playback-Session-Id” value.

About range request field for tracked context per session:

Note: If range value is specified to be null (or not specified) for particular context per session, control plane will increment the value per HTTP request to create a distinction between each HTTP request.

Example 2:

```
    - type: 'video'
      target: 'hulu'
      options: 'cache'
      hostsignature: '"akamaihd.net";"hulu.com"'
      httprangefield: 'Range'
      httpsessionfield: 'X-Playback-Session-Id'
      url:
            - signature: '/hulu[0-9]'
              maxmatchsize: 10
              cachekeyid: '[^\/]+\/[^\/]+\/([^\/]+)\/'
              contextid: '[\&\?]authToken=([^\&]+)'
              cachekeymisc: '[^\/]+\/([^\/]+)\/[^\/]+\/([^\/\?]+)'
```

The same as advanced configurations example1 except that no relationships needed to be established with multiple request patterns.