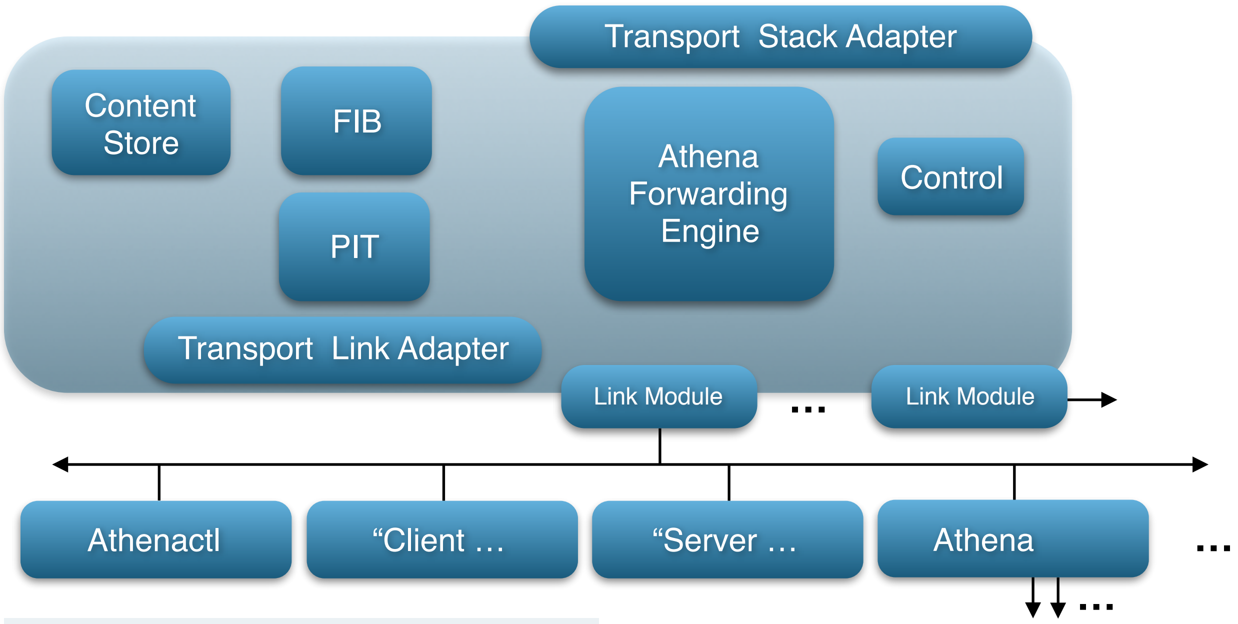
Networking and Distributed Systems
Computing Science Laboratory
Copyright 2015-2016

CCNx Athena Users Guide

Kevin Fox[1*]

Abstract
Users guide to configuring and using Athena and its utilities.

[1]*Computing Science Laboratory, PARC*
*__Corresponding author__: kevin.fox@parc.com



1. Athena

Athena is a prototyping framework for testing and developing CCNx forwarding agents. Its runtime elements include a forwarding agent, **athena**, and a command line interface for submitting interest control messages, **athenactl**. CCNx 1.0 source also includes several example scripts for composing topologies for CCNx simulations using **athena** and **athenactl**.

1.1 Athena Forwarder

The Athena forwarding agent (**athena**) is the main executable which provides CCNx forwarding capabilities. It requires at least one initially configured link in order to interact with messages on the network, initially those being sent as interest control messages to **athena** to solicit further configuration requests.

Links used by **athena** represent a pair of connected endpoints that can be supported by any number of protocols, by default Athena provides link modules for tunneling over tcp and udp (IPv4) as well as configuring native ethernet. Link modules are provided as shared libraries, allowing Athena to be extended to support any number of arbitrary transport mechanism as long as the referenced link module is available at runtime.

At startup, if no other command line link parameters are provided, **athena** creates a pair of links, a localhost tcp listener and tcp listener on its primary IP address, both on port 9695. There are no additional advertisements or interface probes performed by the forwarder by default. Command line connection parameters will override these default connections.

Once an initial link has been configured it can be connected to by any external agent that is capable of sending interest control messages on that transport, to allow modification of the configuration and behavior of the forwarder (e.g. **athenactl**).

The **athena** command line options are:

> **-c (—connect) <link specification>**
> **-s (—store) <value>**
> **-o (—statefile) <filename>**
> **-i (—config) <filename>**
> **-v (—version)**
> **-d (—debug)**
> **-h (—help)**

The connect flag is used to specify a non-default link for the forwarder. It may occur multiple times on the command line and each link specification is created in the order listed.

The Athena content store is disabled by default. Providing a capacity value for the store defines the size of the content store, in MBs, that Athena will allocate. For instance, "**athena -s 1** .." will start an Athena instance with a 1 megabyte content store.

The state file parameter gives the name of a file to which **athena** configuration changes will be written to. Those same file contents can be given to **athenactl** to resubmit those commands to **athena**, or given directly to **athena** with the -i option.

A debug flag turns on **athena** debug logging. Logging levels, can also be set using **athenactl**.

2. Link Specifications

Link specifications are used to define protocol specific links.

A link specification provides the name of the link module to use in its schema (tcp, udp, eth), followed by link specific parameters and flags. Athena currently provides **TCP**/**UDP** tunnels, as well as a native **ETH**ernet transport link module. Alternate modules can be provided to change or extend support to any other type of message transport (e.g. IPv6, shared memory, …).

Link specifications are formatted as URI strings with the schema specifying the link module to be loaded, an "authority" that provides link specific address information, and the remainder of the path specifying flags and options that are specific to the link module.

2.1 TCP link module

```
tcp://<IPv4>:<port>/<options …
        /listener
        /name=<linkName>
        /local={true|false}
```

2.2 UDP link module

```
udp://<IPv4>:<port>/<options …
        /listener
        /name=<linkName>
        /local={true|false}
        /src=<sourceIPv4address>
        /mtu=<mtuSize>
```

2.3 ETH link module

```
eth://<device>:<address>/<options …
          /listener
          /name=<linkName>
          /local={true|false}
          /src=<sourceMACaddress>
          /mtu=<mtuSize>
          /fragmenter=<fragmenterType>
```

2.4 Link module options

Although options are interpreted specifically by the link modules they are applied to, some options are common and have the same, or similar, semantics across modules.

2.4.1 TCP link module options

- /listener

Specify that a listener should be instantiated on the link. Messages are not routed over listener links, only new connections can be accepted and will be managed by the instance that's created. For UDP and Ethernet links messages are multiplexed by the listener to queues owned by their associated links.

- /name=<linkName>

Provide a name that the link will be referenced by in other operations (like adding routes). By default, the link module creates a locally unique name for the link, typically composed of the source and destination addresses. Any specified name must be unique within the forwarder and will be used instead of the default assigned by **athena**.

- /local={true|false}

Force a link to be treated as though it were remote (local=false) or local (local=true). By default, the forwarder determines if the link is local or not by checking if it only connects locally, that is, the link doesn't connect to another machine (see section 3).

2.4.2 UDP link module specific options

- /listener
- /name=<linkName>
- /local={true|false}

(See section 2.4.1)

- /src=<address>

The source address that the connection should come from. This is can be used to create point-to-point links where neither side is a listener but each must know the others origin. If the src pragma is not given, the operating system will pick an IP/Port that's available but one that you can't specify to the other side of the link without knowing about it beforehand.

- /mtu=<mtuSize>

Specify a link MTU size. For UDP this typically defaults to 64k. By specifying an MTU size, messages which exceed it will fail to be sent, issuing an EMSGSIZE error which will cause Athena to drop the message.

- /fragmenter=<fragmentationType>

Specify the fragmentation protocol to use.

BEFS: Begin End Fragmentation Scheme
http://www.ccnx.org/pubs/icnrg_fragmentation_150719.pdf

2.4.3 ETH link module specific options

- /listener
- /name=<linkName>
- /local={true|false}

(See section 2.4.1)

- /src=<sourceMACaddress>

Specify the source MAC address that the connection should use. This is primarily used to create point-to-point links where neither side is a listener.

- /mtu=<mtuSize>

Specify the link MTU size. This defaults to the configured device MTU size (typically 1500). By specifying the MTU size, messages which exceed the MTU will fail to be sent, issuing an EMSGSIZE error which will cause Athena to drop the message.

- /fragmenter=<fragmentationType>

Specify the fragmentation protocol to use.

BEFS: Begin End Fragmentation Scheme
http://www.ccnx.org/pubs/icnrg_fragmentation_150719.pdf

2.5 Link Specification Examples

A localhost TCP listener on port 5555:

- tcp://localhost:5555/listener

A link to a TCP listener on port 5555:

- tcp://localhost:5555

In the above examples, if both links were created on the same a**thena** instance a loopback connection would effectively be created.

In all cases where a IP address is *localhost*, or associated with a local interface, links are considered local and the forwarder treats them as such. That is, hop limit fields in messages are not decremented when they traverse a local link.

By using a local=false flag, the forwarder link module will treat the link as remote and the forwarder will decrement message hop limits as they traverse the link.

- udp://localhost:5555/listener/local=false

The following specifies a UDP link to the above "remote" listener:

- udp://localhost:5555/local=false

Both ends of the link must be provided the same local flag in order for link behavior to be consistent.

A remote link can also be forced to be local by specifying "local=true".

The local flag is primarily meant for use in simulations where explicit semantics of the link are necessary and the forwarder internal determination (whether a link is local or remote) must be overridden in order to perform the expected hop limit operations. For the simulation of an environment where a pair of forwarders reside on the same host, a local=false flag will force the link to act as though both were remote to each other even though that is not the case.

NB: a connection between forwarders using the public IP addresses of the same node are, by default, marked local as the forwarder also assumes that if the source and destination addresses match it's a local connection. In this case, to force non-local semantics the local=false pragma needs to be included in the link specification on both endpoints.

3. Athenactl

3.1 Athenactl

Athenactl provides a method to send interest control messages to an athena forwarding agent. Interest control messages can be sent from any process that can create an authenticated connection and send valid interest control messages.

Athenactl can be used to augment, inspect and/or change the configuration and behavior of an Athena forwarder instance. Its command line flags specify the key store and password to be used to authenticate the user, an optional link name to connect to the forwarder with and the intended command with its arguments.

• **athenactl** -f <identity file> -p <password> <cmd…>

By default, athenactl uses tcp://localhost:9695 to connect and interact with the forwarder, however, it can be provided with an optional link specification to connect to a specific forwarder interface. Currently, only tcp links are supported.

• **athenactl** -a tcp://<host>:<port> …

You can provide **athenactl** with a file that contains a set of ccnx: commands and arguments to execute, such as those created by the -o option of **athena**.

• **athenactl** -i <file>

What follows are the current set of commands that are supported by **athenactl**:

add, list, remove, set, unset, spawn, quit

• **add**

- **link**
  - **<link specification>**

  Add a new link using the specified URI.

- **route**
  - **<link name> <prefix>**

  Add a route to an LCI via the specified link.

• **list**

- **links**

  List the currently configured links.

- **fib**

  List the current fib contents.

• **remove**

- **link**
  - **<link name>**

  Remove a link by name.

- **route**
  - **<link name> <prefix>**

  Remove the specified link as a prefix route.

• **set**

- **debug**

  Set the log level to debug for notifications.

- **level**
  - **debug | info | off | all | error | notice**

  Set the log level for forwarder notifications.

• **unset**

- **debug**

  UnSet the debug log level for notifications.

• **spawn**

- **<port> | <link specification>**

  Spawn a new local forwarder instance with the specified port or link.

• **quit**

  Force the forwarder instance to exit.

• **ccnx:<command> <args>**

  Give athena an explicit URI command and payload.

3.2 Athenactl Examples

[3.2.1] Create a TCP link named C1 from an initial athena instance to a second athena instance listening on port 5555, with a route setup for lci:/tut to the instance at port 5555.

A. **athena &**
B. **athenactl spawn 5555**
C. **athenactl add link tcp://localhost:5555/name=C1**
D. **athenactl add route C1 lci:/tut**
E. **athenactl remove route C1 lci:/tut**
F. **athenactl remove link C1**

G. **athenactl -a tcp://localhost:5555 quit**
H. **athenactl quit**

Step (A) backgrounds an initial Athena instance, which is listening on the default port (9695), then uses athenactl to spawn a second instance that's listening on port 5555 (B). A link named C1 is created between the two forwarder instances in (C), and in (D) we add a local route for lci:/tut is created on the 9695 instance to the 5555 instance via link C1.

In (E), start the process of tearing down the configuration, where the C1 route to lci:/tut is removed and step (F) removes the C1 link itself, detaching the two Athena instances. The removal of the route in step (E) could potentially be skipped as removing the link in (F) will scrub all references that use link C1 in both the PIT and the FIB, implicitly removing the route. (G) forces the spawned instance listening on 5555 to exit and (H) forces the original backgrounded **athena** instance to exit.

[3.2.2] Create a TCP tunnel between two **athena** instances.

A. **athena &**
B. **athenactl spawn 9696**
C. **athenactl add link tcp://localhost:9696**
D. **athenactl list links**
E. **athenactl -a tcp://localhost:9696 list links**
F. **athenactl -a tcp://localhost:9695 quit**
G. **athenactl -a tcp://localhost:9696 quit**

Create two forwarder instances, one listening on the default port (9695) and the second listening on port 9696. (C) adds a link from the forwarder on port 9695 to the forwarder on port 9696. (D) lists the links configured on the 9695 instance and (E) lists them on the 9696 instance so you can verify the link from both end points. (F) and (G) force the two instances to exit, implicitly closing the link.

[3.2.3] Create a UDP point to point tunnel.

A. **athena -c tcp://localhost:5000 &**
B. **athenactl -a tcp://localhost:5000 spawn 5001**
C. **athenactl -a tcp://localhost:5000 add link udp://localhost:5001/src=localhost:5000/local=false/name=N1toN2**
D. **athenactl -a tcp://localhost:5001 add link udp://localhost:5000/src=localhost:5001/local=false/name=N2toN1**

Create a pair of forwarder instances, the initial forwarder (A) listening on port 5000, the second (B) listening on port 5001. Since this is not using the default port the address of the forwarder must be specified when running athenactl, using the -a <link specification> argument. (B) connects to the initial instance at port 5000 and has it spawn the second instance at port 5001. (C) and (D) setup a pair of symmetrical UDP links. In (C) connect to the forwarder at port 5000 and add a link named N1toN2 pointing to the forwarder at port 5001. (D) connects to the forwarder at port 5001 and create a link named N2toN1 that connects with the forwarder at port 5000.

Both link specifications provide a source port to use on their link instance to establish the proper endpoint to use locally. This creates a point-to-point tunnel topology, no messages have yet to traversed the link.

The "local=false" directive forces the forwarder to treat the link as remote so messages which traverse the link will have their hop limit counts decremented appropriately.

[3.2.4] Add a route to prefix '/foo' on a link named bar

- **athenactl add link tcp://localhost:9695/name=bar**
- **athenactl add route bar lci:/foo**

[3.2.5] Listen to 192.168.1.7 on tcp port 9695

- **athenactl add link tcp://192.168.1.7:9695/listener**

[3.2.6] Listen on interface en0

- **athenactl add link eth://en0/listener**

[3.2.7] Create a udp connection to 1.1.1.1 port 1200

- **athenactl add link udp://1.1.1.1:1200**

[3.2.8] Create a udp connection to 1.1.1.1 on port 1200 from the local address 2.2.2.2 port 1300

- **athenactl add link udp://1.1.1.1:1200/src=2.2.2.2:1300**

[3.2.9] Create a udp connection to ccn.parc.com on port 9695

- **athenactl add link udp://ccn.parc.com:9695**

[3.2.10] Create an ethernet connection on em3 with a specified source mac address (the configured address for em3 is used if the mac isn't specified)

- **athenactl add link eth://em3:e8-06-88-cd-28-de**

[3.2.11] Create an ethernet connection on eth0 using a named MAC from the local ethers file

- **athenactl add link eth://eth0:virtualBox-mac**

[3.2.12] Create an ethernet connection on eth0 using BEFS fragmentation protocol.

- **athenactl add link eth://eth0/fragmenter=BEFS**

[3.2.12] Create an UDP connection using BEFS fragmentation protocol.

- **athenactl add link udp://localhost:9695/fragmenter=BEFS**

3.3 Example Scripts

The CCNx distribution athena command-line directory contains scripts for creating sample simulated topologies:

• **startTutorial.sh**

Creates a ring of twenty forwarders along with a circular route for lci:/ccnx/tutorial. Interest messages sent to lci:/ccnx/tutorial will traverse the entire ring until their hop limit is exceeded. An instance of ccnxSimpleFileTransfer_Server is started so ccnxSimpleFileTransfer_Client can then be used to transfer files across the topology.

• **startMesh.sh**

Create a fully connected mesh of four forwarders, with no routes or services configured.

• **athenaTopologyGen.py**

A python script that can parse dot files and create a running topology based on its specification. An **athena.dot** file is provided as an example.