Intra- and Inter-System Communication

This document lists the different communication channels between nodes within a router, between peering routers, and between routers and their conductor. This is to provide:

- A guide for port forwarding when deploying a conductor behind a firewall
- A tool to help predict and manage the amount of ambient, management-plane bandwidth that 128T software will use during "steady state." (This is useful when deploying devices that have limited use and/or expensive WAN connections, such as LTE.)

Connections

Each running instance of 128T software (generically termed as a *node*) can exist in one of two *roles*: a *router* or a *conductor*. Furthermore, two *nodes* can be coupled together to form a highly available router or conductor. There are machine to machine (M2M) connections between the various topologies of nodes, which will be covered here. These include:

- Node to node connectivity within a highly available router
- Router to router connectivity between peering routers
- Router to conductor connectivity for management, adminstration, and orchestration

Node to Node Connectivity (High Availability)

The vast majority of node-to-node connections, deployed as highly available systems, are done over directly connected interfaces between collocated machines. As such, bandwidth consumption and firewall management are not applicable. However, it is possible to geographically separate highly available nodes and have them communicate over a WAN, typically in the form of a data center interconnection.

> Note: geographic distribution for highly available systems is only supported for nodes serving as conductors. 128 Technology does not support geographic distribution of nodes acting as routers.

When deploying geographically separated conductor nodes, the following requirements must be met:

- Latency less than 150ms
- Packet loss less than 0.01%
- No firewalling between the systems

Router to Router Connectivity

Between peered routers, there are four different M2M connections that are established aside from forwarding plane traffic that is sent between them, and excluding routing protocol control plane traffic (BGP). Each of the connections is described here.

> Note: Peering 128T routers require a successful exchange of BFD packets in order to initiate SVR connections. *At least one of the devices must be reachable at 1280/UDP.* Peering can be established if one device is behind a NAT that allocates dynamic ports, but not both. To peer devices that are both behind distinct NATs (not recommended), you must forward 1280/UDP to one or both of them.

BFD (Peer Path Detection)

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	1280/UDP	87	89	1s	Frequency negotiated between devices by configuring `desired-tx-interval` and `required-min-rx-interval` settings.

BFD, or Bidirectional Forwarding Detection (as defined in RFC 5880), is exchanged between 128T routers to detect SVR path availability. I.e., the successful, and continued exchange of BFD packets is a prerequisite for choosing that path to carry SVR traffic. For peer path detection, 128T uses the *asynchronous mode* of BFD.

The peering status between 128T devices is viewable using the PCLI command `show peers`.

The default timer values for BFD traffic are adequate for deployments where head end routers are managing 1,000 aggregate peer paths. For larger deployments, these values should be dilated.

BFD (Path Quality)

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	1280/UDP	840	880	10s	A ten (10) packet burst sent between 128T routers on each peer path.

Each 128T router will measure the path quality to its peer using the *echo mode* of BFD. It does so by sending a burst of ten packets every ten seconds over each peer path. These packets are echoed back from the remote 128T instance and returned to the point of origin. This allows the transmitter to accurately measure round trip time, and calculate packet loss and jitter (variability in interpacket arrival times).

The results of the path quality testing are available in the output of the PCLI command `show peers detail`.

> Note: the 128T will also *calculate* a Mean Opinion Score (MOS) for each peer path. This is derived from the loss, latency, and jitter values empirically determined by the Path Quality BFD packets.

In many common deployment scenarios, the 128T software is deployed as "hub-and-spoke," where the traffic flows exclusively (or nearly exclusively) in one direction from branch deployments toward data centers. In these topologies, it is common and recommended to disable the asynchronous packets sent by the hub routers toward the edge routers, since the path selection critieria is not relevant.

Firewall Detector

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	1280/TCP	6516	6516	300s	Rate can be adjusted by setting `udp-transform/detect-interval` or disabled by setting `udp-transform/mode` to `always-transform`.

Each 128T periodically sends *firewall detector* packets on each peer path to determine if stateful firewalls exist on the peer path. Firewalls between 128T devices can interfere with SVR behavior; because of this, the firewall detector is used to automatically trigger a *UDP transform* feature, to carry SVR over UDP when firewalls would block TCP. The status of the firewall detector is show in the PCLI in the output of `show udp-transform`:

```
admin@labsystem1.fiedler# show udp-transform router newton
============= ============ ============ ==========
=========================================
 Router Name   Node Name    Peer         Status     Reason(s)
============= ============ ============ ==========
=========================================
 newton        labsystem2   becket       enabled    TCP SYN; Mid-flow; TCP SYN
Jumbo;
                            becket       enabled    TCP SYN; TCP SYN Jumbo;
                            burlington   enabled    TCP SYN; Mid-flow; TCP SYN
Jumbo;
```

The firewall detector function sends deliberately malformed, out-of-order, and missequenced packets in bursts of four packets every five minutes (by default). This time may be dilated if you are assured there is no firewall on the path, nor any possibility of the path changing to include a stateful device at any point in the future. If you know there is a firewall device in the path, you can force the `udp-transform` to `always-transform`, which suppresses the firewall detector packets entirely.

Path MTU Discovery

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	1280/UDP	2945	90	600s	Interval is configurable within `path-mtu-discovery/interval`, or disabled `path-mtu-discovery/enabled`.

Peering 128T routers will perform path MTU discovery on each peer path between each other. This test is run every ten (10) minutes by default, to adjust in the event of path changes between peering devices.

The discovered MTU is viewable in the output of `show peers`.

Router to Conductor Connectivity

Each deployed router (and in many cases, individual nodes within that router) has multiple concurrent connections to each conductor node within its authority. The primary connection between a router and a conductor is using 930/TCP, which is an encrypted SSH connection that bears most router-to-conductor inter-process communication (IPC). The secondary connetion is that between a router's *salt-minion* and a conductor's *salt-master*, which leverages 4505-4506/TCP.

> Important: when deploying a firewall in front of your 128T conductor, it is important to ensure that ports 930/TCP, 4505/TCP, and 4506/TCP are forwarded to your conductor node.

Software Version Check

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
to router	4506/TCP	950	2100	12hr	The payload size will be variable based on how many software versions are available for upgrade.

Every twelve (12) hours, the conductor will query the router to see which versions it is eligible to download, to render in the various user interfaces (PCLI, GUI). This value is not configurable.

Alarms

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
to conductor	930/TCP	214 (est.)	150	variable	The payload size and frequency of alarms sent from a router to a conductor is variable based on the types and frequencies of alarms.

Alarms are sent by every managed router node to the conductor using a secure socket. The payload exchanged will be entirely dependent on the frequency and count of alarms that need to be sent.

Alarms are shown on the GUI of the conductor as well as via the PCLI command `show alarms`.

Site-Wide Entitlement

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
to conductor	930/TCP	182 (est.)	0	300s	The payload size is variable based on the name of the device transmitting the entitlement data.

Each router relays its utilization data to the conductor every five minutes, to render it on the conductor UI as part of the conductor's *entitlement* graph. This value is not configurable.

The current entitlement usage is shown in the device GUI as well as the output of the PCLI command `show entitlements`.

Conductor/Asset Keepalives

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
to conductor	4505/TCP	0	0	10s (client), 20s (server)	TCP keepalive ACKs only

The TCP socket between the router node's *salt-minion* and the conductor's *salt-master* is refreshed periodically to ensure that the minion/master connectivity persists. This value is not configurable.

Node Info

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
to conductor	930/TCP	311 (est.)	312 (est.)	15s	Variable payload size based on router/node names.

Key Exchange/Security Rekeying

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
to conductor	930/TCP	150	294	none	Variable based on configured `rekey-interval` and the number of `security` configuration elements.

The conductor orchestrates a periodic key exchange when the `authority > rekey-interval` is changed from its default value of `never`. When set, each router will receive a key from conductor at the defined interval for each `security` element configured.

The status for the key exchange and rekeying process is shown in the output of the PCLI command `show security key-status`.

Asset Keepalives

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	4506/TCP	514	84	1s	

The asset keepalive messages are used to ensure persistent connectivity between a router's *minion* and the corresponding *master* running on its conductor(s). The asset's connection state is viewable using the `show assets` command within the PCLI on the conductor.

Shell `connect`

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	930/TCP	104	104	20s (client), 5s (server)	Amount of data exchanged is largely dependent on administrator activity once connected to a remote router.

The conductor offers several ways of connecting to a remote router's shell (including both its PCLI shell and Linux shell): either via the `connect` command within the PCLI, or using the *PCLI cut-through* feature within the conductor's GUI. Both of these leverage a secure (SSH) connection between the router its conductor(s).

Each *node* within a router will connect to each of its conductors.

Log Retriever Connection

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	930/TCP	104	104	20s (client), 5s (server)	Amount of data exchanged is entirely dependent on if/when administrators retrieve logs from remote routers via conductor GUI.

The *log retriever* is a function within the conductor's GUI to allow administrators to download log files, packet captures, and tech-support-info bundles. When using this GUI function on the conductor, it will retrieve a file inventory via this connection, and also use that same connection to relay any logs to the user via their web browser.

Each *node* within a router will connect using this mechanism to each of its conductors.

Conductor-hosted Software Repository

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	930/TCP	104	104	20s (client), 5s (server)	Only exchanged if the conductor hosted software repository is configured.

When the conductor is used as a software repository for its managed routers, there will be connectivity checks sent periodically. The frequency at which these are exchanged is not configurable. When the conductor is not used as a software repository (which is the default behavior), these keepalives are not sent.

Conductor Software Proxy

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	930/TCP	104	104	20s (client), 5s (server)	Only exchanged if the conductor is configured as a proxy for software retrieval.

When the conductor is used as a proxy for reaching the 128T software repository, there will be connectivity checks sent periodically. The frequency at which these are exchanged is not configurable. When the conductor is not used as a software proxy (which is the default behavior), these keepalives are not sent.

SSH Keepalives

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	930/TCP	84	48	5s	Maintains secure socket connection between conductor and deployed router.

The connection between the router and conductor is refreshed every five seconds using keepalive messages.

Dynamic Peer Update

Direction	Port/Proto	Client Payload (bytes)	Server Payload (bytes)	Default Interval	Notes
bidirectional	930/TCP	128	100	on demand	Variable size payload based on router and node name. This is exercised whenever a dynamic IP address on a deployed router changes.

The Dynamic Peer Update (DPU) process supplies dynamic IP address information from a router to conductor, which will then propagate that information down to all routers that create adjacencies to that address. When a `network-interface` that uses dynamic addressing (e.g., DHCP, PPPoE) acquires an address for the first time or changes its address, that information is sent to conductor via DPU.

The status of the DPU exchange is shown in the output of the PCLI command `show dynamic-peer-update`.