Routing Heterogeneous CCI Subnets

Scott Atchley
Technology Integration Group, NCCS
Oak Ridge National Laboratory
Oak Ridge, TN, USA
atchleyes@ornl.gov

Abstract—**This electronic document is a "live" template and already defines the components of your paper [title, text, heads, etc.] in its style sheet.**

Keywords—Common Communication Interface; CCI; routing; networking; heteroneneous; overlay networks

I. Introduction

This design document describes something.

II. Motivation

Networking technologies for use in local-area networks (LAN) have improved performance (e.g. reduced latency and/or increased throughput) as well as reduced the workload on the host's processors. Wide-area networking (WAN) still relies mostly on Sockets using TCP or UDP. The downside to Sockets is that it does not lend itself to these advances in networking technology.

III. Definitions

In order to discuss the various goals, requirements, and the design, we need to define some frequently used terms:

A. Transport

A transport is a combination of networking hardware and software that provides end-to-end communication. It may use generic hardware and software (e.g. Ethernet and Sockets) or specialty hardware and software (e.g. InfiniBand and Verbs).

B. Subnet

For the purposes of this document, a sub-network (i.e. subnet) is a collection of networked hosts that are visible to one another using a given transport.

C. Autonomous System

An Autonomous System (AS) is an independent organization, which determines its internal routing topology and policy. Each AS terminates at the edge of the wide-area network (WAN). We use the terms AS and organization interchangeably.

D. WAN

Wide-area network. WANs may be public (shared) or private (dedicated) networks.

E. Peering

Routing between Autonomous Systems over the WAN.

F. Router

A router is a process that connects to two or more subnets (within an AS or between two AS) and provides communication between them.

G. Endpoint

In CCI, an endpoint is a process' virtual instance of a network interface card (NIC). The endpoint is the source and destination of all network traffic. An endpoint is per process, not per peer (i.e. a single endpoint can communicate with any number of peer endpoints).

H. Connection

A CCI connection denotes the ability of two CCI endpoints to communicate. A routed connection is the end-to-end connection that spans two or more transport connections.

I. OS-Bypass

OS-bypass is the ability of a process to directly access network hardware without going through the kernel. These accesses may include data movement or status updates (e.g. events).

J. Zero-Copy

Zero-copy is the ability to move data directly between a user-space application and a network adapter without intermediate copies, especially copying to/from the kernel.

K. Messages/Messaging

CCI's Messages (MSG) is a method of communication using a small, unexpected message semantic. Applications can use MSGs for control messages and for bulk data movement rendezvous protocols.

L. Remote Memory Access

CCI's Remote Memory Access (RMA) is a method of communication using a one-sided semantic. RMA is designed for bulk data movement.

With certain transports, CCI can use zero-copy and OS-bypass to improve performance. With other transports (notably, Sockets), CCI implements RMA in software and cannot use zero-copy or OS-bypass.

M. RMA Completion Message

Because a RMA is one-sided operation, a process may need to alert the peer process when the RMA operation is complete. CCI provides the ability to send a MSG when a RMA completes to notify the remote process.

IV. Goals and Requirements

We identified the following goals and requirements:

A. Goals

The primary goal for CCI routing is to provide end-to-end communication over heterogeneous networks across local-area networks and wide-area networks. One usage scenario is moving data from a simulation on a leadership class system across the local-area network, over the wide-area network to another DOE facility, across its local-area network, and finally into a cluster for analysis and/or visualization. Such a scenario could transit five or more heterogeneous networks: a high-performance interconnect within the leadership class system, the local-area network, the wide-area network, the second facility's local-area network, and the cluster's high-performance interconnect.

The secondary goal for CCI routing is to take advantage of the highest performing networking stack on each network. We could simply use Sockets and take advantage of the routing capabilities of the IP stack. Because of the design of Sockets precludes OS-bypass and zero-copy techniques, we need to use the non-Sockets APIs for the networks that provide those capabilities. CCI provides the ability to exploit each network's capabilities, but CCI does not provide by itself a common address space for routing.

Our last goal is to provide that common address space to enable routing with minimal impact on the current CCI API and implementations. Ideally, current CCI implementations would not need any modifications to support routing.

B. Requirements

We have identified the following requirements:

1) Route between heterogeneous networks

CCI currently supports multiple networks including Sockets/UDP, Sockets/TCP, Verbs/InfiniBand, GNI/Gemini, and Portals/SeaStar with additional transports under development by collaborators.

2) Route between organizations

CCI routing needs to support routing between different organizations (i.e. distinct administrative domains).

3) Support routing to/from private sub-networks

Most of the CCI TransPorts (CTP) use an IPv4 address as part of the name for an endpoint, but typically these are private, non-routable addresses and the address is used for connection setup only. Each networking stack has unique addressing conventions for the underlying communication and do not lend themselves to a common addressing scheme.

4) Support multiple routers between networks

Allowing multiple routers enables improved throughput by aggregating bandwidth, improved fault-tolerance by eliminating a single point-of-failure, and reduced congestion by load balancing over multiple links.

5) Support multiple metrics to determine the best route

The routing design should allow organizations to determine the routing policy best suited to the local needs. These might include shortest path (i.e. fewest hops), network maximal throughput (i.e. choose higher bandwidth networks over lower bandwidth networks), network hardware capabilities (i.e. favor networks that provide hardware/software acceleration over traditional Sockets), etc.

6) Provide the ability to set static routes

Static routes are useful for testing as well as for ensuring the use of dedicated resources.

7) Minimal modifications to the CCI programming interface

Applications should not need to be rewritten to use CCI routing.

8) Minimal modifications to existing CTP implementations

The CCI TransPorts should not need to be modified. This requirement assumes that routing will use the CTPs as they currently exist and implement the routing protocols on top of the CTPs.

9) Routers need to run in user-space

All current implementations of CCI only provide user-space libraries. Organization policy might require that some routers be run in privileged mode (i.e. a user-space process run as root or other special user).

C. Non-requirements

Equally important to the design is to identify non-requirements (i.e. items outside of the scope of the project).

1) Router auto-discovery

For traditional routable networks (i.e. Ethernet), routers use broadcast to discover other routers. Clients may also use broadcast to either discover routers or request networking configuration information (e.g. DHCP) that contains router information. Broadcast may be expensive or even not available on some large systems.

2) Internet-sized scaling

This does not mean that routing should not cross the wide-area network. It simply means that CCI routing is an example of overlay routing on top of existing networks, including the Internet. Because the need to scale is less than that of the Internet as a whole, some of the solutions may trade-off scalability for increased performance, for example.

3) Forwarding through an AS

Each organization is an end destination in the overlay network. We do not need to provide forwarding through an AS (e.g. ORNL->ANL->Livermore). Not requiring forwarding is also an example of item 2 above.

A corollary of this non-requirement is that all subnets within an organization are reachable from all other subnets within the organization via one or more network hops. It will not be legal to have two or more subnets connect to the WAN from an organization that cannot route to each other within the organization. If this were allowed, it would require forwarding over the WAN. If an organization has such subnets, the organization could use multiple AS numbers to allow WAN routing between their distinct subnets.

4) Multi-pathing

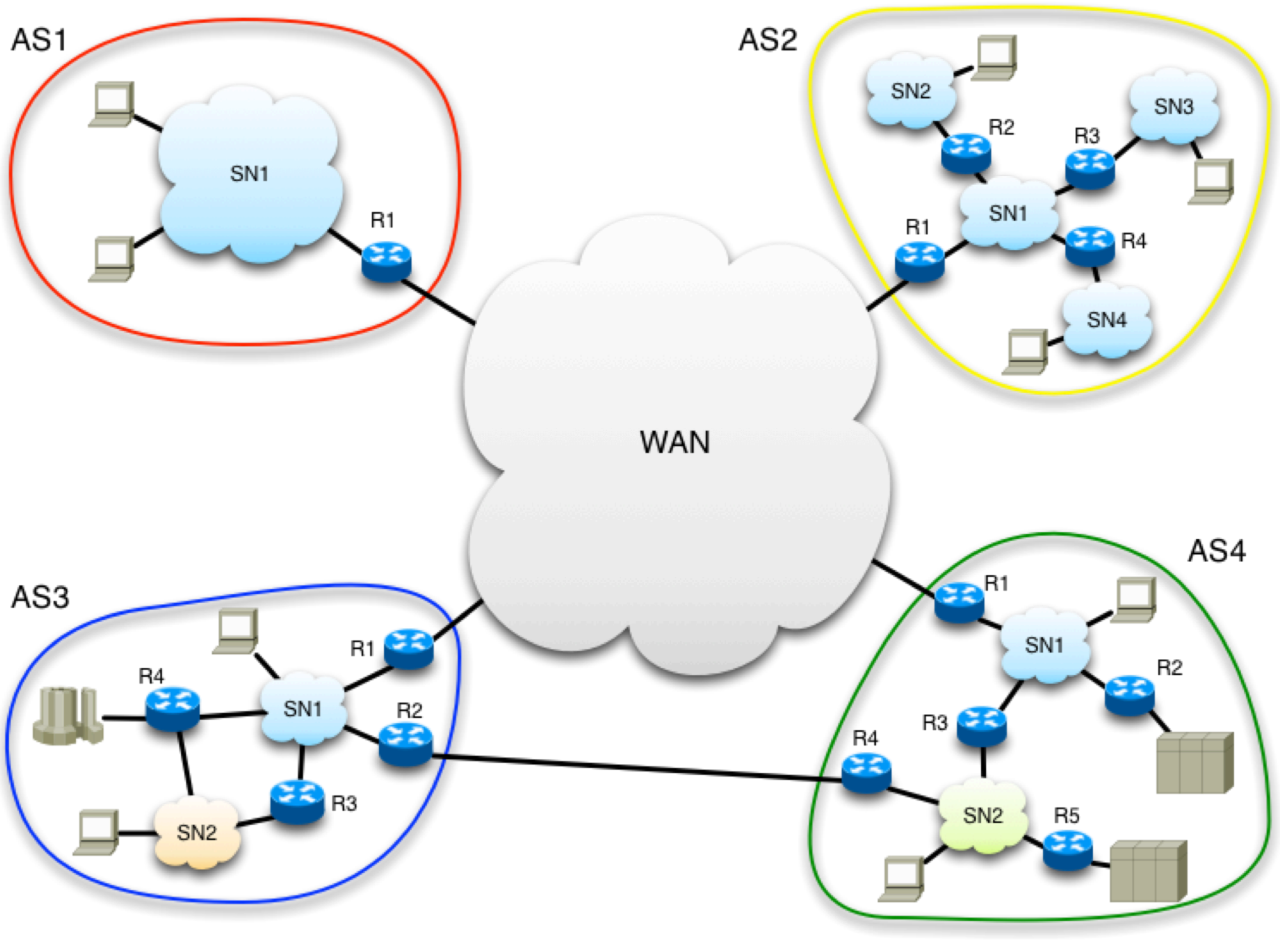

Figure 1. Multiple organzations (i.e. Autonomous Systems) each with multiple subnets.

CCI provides a connection-oriented semantic. During the establishment of a connection, routers may be able to choose from multiple routers for the next hop. Once the connection is established, however, all messages will travel over the selected set of routers. The underlying networks may provide multiple links (e.g. Gemini, bonded Ethernet), but the connections between routers and between routers and hosts will originate and terminate at the same CCI endpoints within a specific connection.

V. CCI Routing Overview

As mentioned in section IV, the goals include routing across heterogeneous networks between distinct organizations. Fig. 1 shows four hypothetical organizations with one or more subnets each and all organizations are connected to the WAN. Two of the organizations, labeled AS3 and AS4, have a dedicated link separate from the public WAN.

For the purposes of routing, each organization determines its routing topology and policy. We use the term Autonomous System (AS) to describe such an organization. Each AS is assigned a unique 32-bit ID. The limited number of anticipated participating organizations should allow manual assignment of AS IDs.

Within an AS, each subnet is also assigned a 32-bit ID. This ID is only unique within the AS. Different organizations may use the same subnet IDs. Therefore, a specific subnet will have a globally unique combination of AS ID and subnet ID.

In Fig. 1, AS1 might represent a campus-wide IP network and thus only it has a single subnet (SN1). It only requires a router at the border between the SN1 and the WAN.

AS2 depicts multiple Ethernet broadcast domains and this organization prefers the Ethernet CTP. Since the Ethernet CTP requires a common Ethernet broadcast domain (EBD), this organization has to provide routers between each EBD.

Fig. 2 looks more closely at AS3. This organization has three subnets: a campus wide Ethernet network (SN1), a storage-area InfiniBand network (SN2), and a leadership class compute system with a high-performance interconnect (SN3). Each subnet in AS3 has one or more routers providing connectivity to other subnets. For example, the compute system’s subnet 3 is connected to subnets 1 and 2 via router 4 (R4). In addition to its WAN connectivity via router 1 (R1), AS3 also has a dedicated WAN link to AS4. This link is only valid for traffic originating or terminating at AS4 and cannot be used for forwarding to other organizations.

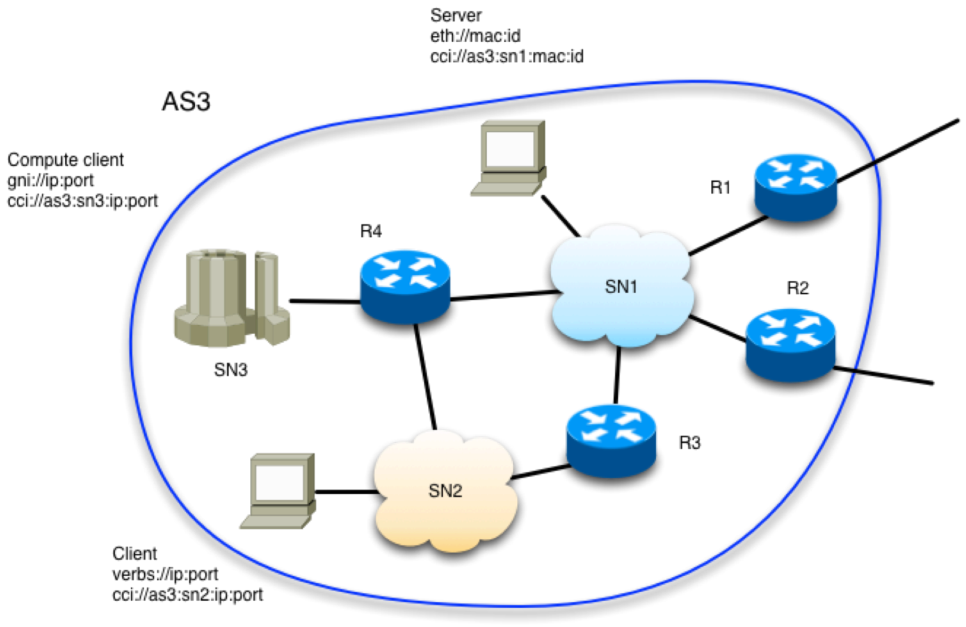

Figure 2. A detailed look at AS3 which has three subnets and four routers.

AS4 has a campus-wide network, a storage-area network, and a couple of HPC systems.

A. *Local Routing*

Local routing is within an organization (or intra-AS). All subnets share the same AS ID. If the AS ID and the subnet ID for two endpoints are the same, the communication does not require routing at all. If the subnets have different subnet IDs, then routing is required over one or more routers within the organization, but not over the WAN.

B. *WAN routing*

If the AS IDs differ between two endpoints, routing over the WAN is required. The routers in one organization do not need, however, the complete (global) routing information for the entire path, as we will see in section VI.

VI. Route Determination

In this section, we will look closer at the details of routing. Each router within an organization will need to have the same route map. The map indicates to which subnets each router connects directly as well as the path from any subnet to every other subnet within the organization. For example, a router that connects to three subnets will have three (or more) network adapters and it will have at least one CCI endpoint per subnet.

Clients of the routing service will never have the map and will not be involved in the building of the routing map. Each client will have a static list of routers available within the device description in its CCI configuration file. The client will randomly choose a router.

Using the organization shown in Fig. 2, we can develop the routing map and given hypothetical network bandwidth for each subnet, we can determine which routes are preferred.

AS3 has three subnets, a public WAN link, and a dedicated WAN link to AS4. Assume that SN1 is a campus-area 10 Gb/s Ethernet broadcast domain. All hosts connected to SN1 can communicate directly with each other. All four routers connect to SN1 and can communicate with each other over SN1. SN2 might be a storage-area network using InfiniBand QDR, which has a data rate of 32 Gb/s. Only routers 3 and 4 connect to SN2. SN3 is within the HPC system and has a throughput of 64 Gb/s. SN3 is only connected to router 4. Lastly, router 2 also has WAN connectivity to AS4 over a dedicated 100 Gb/s Ethernet link.

A. *Building the complete route map*

For this organization, we want to build routes from every subnet to every other subnet. A route will be an ordered list of one or more subnet IDs. For connections between endpoints on the same subnet, no routing is required and the route list is empty.

The routing table can then be thought of as a NxN table where N is the number of subnets. The left column will be the array of originator subnets of connections and the top row will be the array of destination subnets for connections. The intersection of the row N and column M will contain the ordered list of subnet IDs from subnet N to subnet M.

Altogether, AS3 has five subnets (SN1-3, WAN, and WAN to AS4). Its routing table will have five rows by five columns.

We will label the rows and columns 1, 2, 3, W*, and W4 for the five subnets.

1) Local subnet (no routing)

As mentioned previously, if the AS ID and subnet ID match, then we do not need to use routing. Since all routing maps are specific to an AS, the AS ID is the same for all subnets in the map. Therefore, we can identify which entries in the table that will not have any routes.

In our example, the first cell is at row 1 and column 1 for subnet 1 in both cases, which does not require routing and is empty (as are 2:2, 3:3, W*:W*, and W4:W4). Also, since AS3 has two WAN links and since we do not forward through organizations, entries for W*:W4 and W4:W* are empty as well.

2) Single hop routes

When subnets are directly connected via a router, the route uses a single hop. We use them to initialize the routing table.

For example, router 2 connects subnet 1 and subnet 2. For the entry at row 1 and column 2 (1:2), we enter 1,2 and at row 2 and column 1 (2:1), we enter 2,1. We continue with each router's direct connections.

3) Multiple hop routes

Once all the single-hop routes are entered, we need to build routes between non-directly connected subnets. We will build these routes starting with the single-hop routes and combining them until we find all of the possible routes from one subnet to another. To avoid loops when computing routes, if we encounter a subnet ID a second time, we discard the route.

Looking at our previous example, routing from subnet 3 to the WAN could use the following routes:

- 3,1,W*
- 3,2,1,W*

Routing from WAN to subnet could use the reverse of these two routes. No other routes exist from subnet 3 to the WAN without incurring a loop.

In addition to loop detection, we will discard routes that contain a shorter route. In the above example, the route 3,1,W* is contained within 3,2,1,W*. Since no edge (i.e. subnet) can have a zero cost to traverse, the longer route that contains a valid shorter route can never cost less than (or even equal to) the shorter route. Our completed route table for AS3 is shown in Table 1.

TABLE I. ROUTES FOR AS3

	SN1	SN2	SN3	W*	W4
SN1	—	1,2	1,3	1,W*	1,W4
SN2	2,1	—	2,3	2,1,W*	2,1,W4
SN3	3,1	3,2	—	3,1,W*	3,1,W4
W*	W*,1	W*,1,2	W*,1,3	—	—
W4	W4,1	W4,1,2	W4,1,3	—	—

Note that there are no loops since all subnets 1, 2, and 3 are all directly connected to each other. And since we discard longer routes that contain valid shorter routes, the table for AS3 only has one route in each cell.

B. Choosing between multiple routes

If we encounter any loops when building the complete routing table, multiple routes will exist between some of the subnets. Likewise, if an organization has private WAN links in addition to the public WAN link, multiple routes will exist between some subnets and other organizations. The administrator will be able to choose between multiple metrics to determine which route should be used. Initially, we intend to provide bandwidth (based on link-rate, not dynamically available throughput), network capabilities (e.g. native RMA support), latency, and hop count.

When choosing routes, we will use Dijkstra's Algorithm to find the shortest path in a graph. For our usage, each vertex in the graph represents one or more routers that connect two or more subnets. The edges are the subnets. For subnets that cannot be transited (i.e. the subnet can only be at the beginning or end of a route), they are represented by an edge with a vertex only at one end. When dealing with non-transiting subnets, the other vertex will represent the end node.

The algorithm is a minimizing function. The goal is to find the least cost path between any two nodes. At least one of our metrics, bandwidth, needs to be converted. Ideally, the routing algorithm would choose the highest bandwidth network available. In CCI, the device information includes link-rate expressed in bits per second. To convert link-rate to a usable metric, first convert this rate to gigabits per second (Gb/s) and, second, divide a fixed, larger value by the Gb/s. For example, if the fixed value is 1 terabit per second (1 Tb/s or 1,000 Gb/s) and if the device has a link-rate of 10 Gb/s, then the metric for subnet connected to this device would be 100 (1,000/10). For a device capable of 100 Gb/s, the metric would be 10. Given a choice between a subnet with a metric of 100 (10 Gb/s) versus 10 (100 Gb/s), the algorithm will choose the lower value and pick the *faster* 100 Gb/s subnet.

Back to our example, if the application wishes to communicate between on node on AS3's subnet 3 and a node at AS 4, it could use either the public WAN connected to router 1 or the private WAN link connected to router 2. Both routes transit subnet 3 followed by subnet 1.

In our example, subnet 3 has a fast HPC interconnect with a link-rate of 64 Gb/s, subnet 2 is 10 Gb/s Ethernet as is the public WAN, and the private WAN to AS4 is 100 Gb/s. If we convert these, the *bandwidth metric* value for subnet 3 is 15 (rounding down), subnet 1 and the public WAN are 100 each, and the private WAN is 10.

The two routes are then scored. The route through the public WAN traverses subnet 3, subnet 1, and the public WAN for a score of 215 (15 + 100 + 100). The route through the private WAN traverses subnet 3, subnet 1, and then the private WAN for a score of 125 (15 + 100 + 10). The traffic will flow over the private WAN.

This is optimal from AS3's policy, but may not necessarily be optimal from AS4's point of view. For example, if the destination node is directly connected the subnet inside AS4 connected to the public WAN, routing within AS4 may prefer using the public WAN router than the private WAN router. This is another example of a trade-off between optimal routing versus the complexity of ensuring a globally consistent routing map.

Fig. 3 shows a loop within an organization. There are two valid routes from the client to the server. One route traverses subnets 1, 2, and 8 while the other traverse subnets 1, 5, 6, and 8. If the routing metric is hop-count, which is commonly used in traditional IP routing, then the first route will be selected.

If the routing metric is highest bandwidth, we will compute the *cost* of each route using the converted bandwidth. Table 2 shows the link-rate and the metric based on 1 Tb/s divided by the link-rate.

TABLE II. BANDWIDTH ROUTING METRIC

Subnet	Link-rate (Gb/s)	Metric
1	10	100
2	10	100
5	32	31
6	32	31
8	10	100

To determine the cost of a route, we sum the converted link-rate metric for each subnet traversed. The route that uses subnets 1, 2, and 8 has a total cost of 300. The route that passes over subnets 1, 5, 6, and 8 has a total cost of 262 making this the preferred route.

Bandwidth alone does not tell the whole story. CCI supports multiple networks including Sockets (UDP/IP and TCP/IP). Most high-performance networks provide IP or Ethernet encapsulation as well. This means that CCI can communicate over a network using its native low latency, zero-copy, and OS-bypass interface and over Sockets on top of this interface. The native interface will always perform better than the Sockets interface, but the link-rate is identical. In order to recognize this fact, the routing information will include whether the interface supports zero-copy and OS-bypass. When computing the bandwidth metric, we will convert the link-rate to the bandwidth metric and then, if the route does not use zero-copy and OS-bypass, we will double the metric (in effect dividing the link-rate by half). This will bias the routing decision to use subnets with these capabilities over subnets that do not.

VII. IMPLEMENTATION OVERVIEW

In order to provide routing, we will need to make some modifications to CCI and we will have to implement the router application.

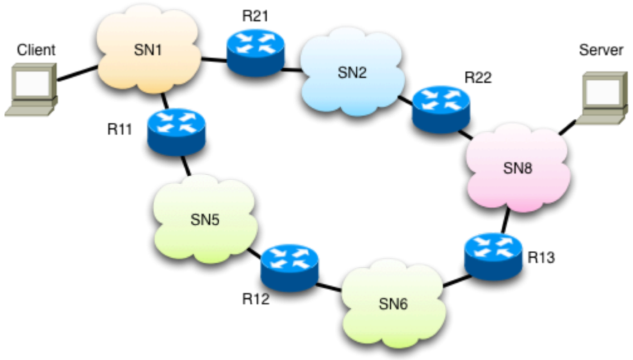

Figure 3. Multiple routes between client and server.

A. *CCI Modifications to Support Routing*

One of the requirements is to make minimal changes to the existing CCI API and transports to support routing. First, we will need to provide address space and router information for each device and, second, we will need to add a new routing transport.

1) *Providing address space information*

Currently, each endpoint can communicate only with other nodes reachable by the underlying network. For the Sockets based transports, these may be private subnets or they may be publicly addressable nodes. For all other transports, they will be private, but possibly quite large, subnets. As described above, we will provide a common address space for CCI using the AS ID and subnet ID pair.

In addition to having the address space information is not enough, each device's endpoints will need to know about the available routers since we will do not require router auto-discovery as mentioned in section IV.

Fortunately, CCI already has a mechanism to pass arbitrary information for each device using the CCI configuration file (i.e. config file). The purpose of the config file is to allow a system administrator to describe which devices should be used and information about the devices. The config file uses a standard INI format with device names and keyword/values pairs for additional information.

The CCI spec only mandates two items when describing a device: a device name and the transport responsible for the device. Transports are free to support additional keyword/value pairs. Transports ignore keywords that they do not understand. Using this feature, we can add three keywords to each device: AS ID, subnet ID, and router.

We will impose some constraints on these keywords. First, all three must be present and must have valid values to be useful for the device. The lack of any of the three keywords or valid values will prevent routing. Second, the AS and subnet keywords must only be specified once per device (i.e. no multiple values). Third, the router keyword must contain a value with a valid URI for the given transport. For example, if the device's transport is Verbs, the URI must be recognizable by the Verbs transport. Fourth, the router keyword *may* be included multiple times to specify the availability of multiple routers.

2) Routing transport

Routing will require managing end-to-end state for connections and communications. Adding support for routing within the current and future transports would require significant changes. Instead, we can avoid modifying current transports completely by implementing a new routing transport that manages this state and uses existing transports for actual communication.

This transport will rely on the underlying transport's send and receive buffers, but it will need to have its own completion queue to indicate when end-to-end events have completed.

3) Endpoint creation

When creating an endpoint, CCI will bind the endpoint to a specific device. Since routing is an overlay on top of actual devices and transports, we will use the flags argument to `cci_create_endpoint()` to indicate that the application wishes to enable routing support. This flag will invoke the routing transport, which will initialize its state and then create an endpoint using the underlying transport. All subsequent CCI calls will then use the routing endpoint, which will pass calls through to the underlying endpoint and manage the end-to-end state as needed.

Without routing, each endpoint has a transport recognizable URI that includes a transport prefix, node identifier, and endpoint id (e.g. port in IP networks). Examples of valid non-routing endpoint URIs include:

- sock://host:port
- verbs://ip_address:port
- eth://mac_address:ep_id.

The routing endpoint will create a URI using a distinct prefix (e.g. cci://), the AS ID, subnet ID, and then the underlying URI's node identifier and endpoint id.

For example, a Cray GNI endpoint URI without routing might be gni://10.101.50.37:6744. If the application passes the routing flag to `cci_create_endpoint()`, the routable URI would be cci://32:126:10.101.50.37:6744, where 32 is the AS ID, 126 is the subnet ID, and the rest is the node identifier and endpoint id.

B. CCI Routers

The routing applications will implement the topology discovery and route selection while using CCI for all communication between routers and between routers and CCI routing clients. A router must have devices on two or more subnets. Routers will not use the CCI routing transport since they only need to manage communications on their local subnets.

Routers must have a CCI config file that specifies which devices to use. If this router needs to provide forwarding over a subnet to another router, the device description for that subnet must include one or more router keywords to determine to which routers it should connect.

In addition to the CCI config file, the router must understand the order of the routing metrics preferred by the local organization. This may be implemented via command line arguments, a config file, etc.

When the router starts, it will open an endpoint on each device in the config file. For each device, it will connect to every router described by the router keyword. When routers connect, they will exchange information about the subnets that they support. A router information record is shown in Table III.

Using these records, they begin to build the network topology. The routers then forward all newly received records to their existing peer routers. This recursive exchange of reachability information permits the routers to build the complete topology as described in section VI. As the topology is built, the routers also compute the forwarding table. If loops are detected, the router computes the preferred route based on the local administrative policy (e.g. bandwidth, latency, hop-count, etc.). Once the router begins building the forwarding table, the router is ready to accept connections from routing clients.

TABLE III. ROUTER INFORMATION RECORD

← 32 bits →

AS ID		
Subnet ID		
Instance		
Rate (Gb/s)	Caps	URI Len
URI		

Although the routers do not use the CCI routing transport, the routers and the routing transport will need to implement a common protocol for connection establishment, handling MSGs ad RMA, and reporting of their completion events.

C. Managing End-to-End State

Both the CCI routing transport and the router daemons will need to manage three types of communication state: connection setup, messages, and remote memory access.

1) Connection setup

CCI uses a three-way handshake when establishing unicast connections. The client initiates a connect call which sends a connect request to the server. The server receives the connect request and chooses to accept or reject it. The response is sent back to the client. Lastly, the client acknowledges the response.

The end-to-end connect must accomplish the same while initiating connections at each hop. The client's routing transport will connect to one of the routers included in the device's `conf_argv` array and send a connect request which includes the final destination. The router will look up the best route and the URI of a router at the next hop. It will then connect to it and forward the destination URI. Each router repeats this until it reaches a router on the last subnet. That router connects to the server. When the server application accepts or rejects, the server's routing transport will send an end-to-end reply back to the client. Lastly, the client will acknowledge the reply.

As with non-routing CCI, the handshake will negotiate the connection's max send size for MSGs, which might be lower than the device max send size for either the client and/or server.

2) *Messages*

Once the connection is established, the two processes can begin exchanging messages. Since CCI establishes connections between peers, the router simply has to associate the two local connections (the one from the client direction and the other in the server direction). Thus, when a MSG arrives on one, it simply forwards to the other without need for any lookup or route computation.

When the last router gets the send completion event, it will send an end-to-end ack back to the client. When the client's routing transport receives this message, it will generate the send completion for the send.

Since the routers will be managing many connections, their shared receive queues may be temporarily busy. It will be important for the underlying transports to handle intermittent receiver-not-ready (RNR) responses until the congestion clears.

3) *Remote Memory Access*

The RMA path is a little more complicated than the MSG path. In general, performing an RMA requires three steps. First, the initiator and target register some local memory. Second, the target passes its RMA handle to the initiator using a MSG. Third, when the initiator has the target's RMA handle, it can then initiate the RMA. The RMA will either *write* (i.e. put) data from the initiators memory into the target's memory or *read* (i.e. get) from the target's memory to the initiator's memory.

The design for routing of RMAs must cope with multiple issues. First, the routers need to register memory before they can participate in a RMA. They will need to register the memory with each device for which they will be forwarding. Second, RMAs may be arbitrarily large (i.e. as large as system memory). Third, routers may not have as much memory as the initiator or the target. Lastly, when forwarding the RMA, the next router may not have enough memory available to continue the RMA, especially if it is attempting to initiate an RMA to the current router.

To address these issues, the router daemons will need to do the following. When each router starts, it will need to register memory for use in forwarding RMAs. To address the issue where the RMA length is larger than the router's registered memory as well as the need to service multiple RMAs from different clients, routers will fragment RMA requests to a consistent fragment size (e.g. 1 MB or 4 MB). Routers will allow multiple fragments from the same RMA request to progress if there is enough memory available, but each router will reserve one fragment for each peer router. The reserved fragment will prevent deadlock in the case where two routers need to exchange RMAs, but have no memory available, which would cause the RMAs to timeout and fail. The routers will need to negotiate the fragment size when they establish their inter-router connections.

As part of the routing protocol shared between the routing transport and the router daemons, RMA writes and reads will be modified.

a) *RMA write protocol*

The initiator's routing transport will not know where to put the RMA write data in the router's memory space. Since only the router will know which memory fragments are available for forwarding, the router will need to read the data from the initiator.

To minimize state on the router, however, the initiator's routing transport will manage the fragmentation of the RMA and send *RMA write requests* to the router for each fragment. The router's preferred fragment size can be sent to the client during the connection handshake. When the router has one or more fragment-sized memory buffers available, it will RMA read the data from the initiator. The router then sends a *RMA write request* for each fragment to the next router along the path. Each router continues to forward the fragments to the next router. When the last router receives a fragment, it will RMA write the data to the target's memory space, thus maintaining the one-sided (i.e. passive) semantics from the target's perspective. The last router will be responsible for sending an end-to-end ack back to the initiator to indicate that the fragment was delivered. The ack will need a cookie that was sent in the *RMA write request* to indicate which fragment was completed. Once all fragments are delivered, the initiator's routing transport will generate the RMA completion event.

If the RMA includes a completion MSG, the initiator's routing transport send the completion message using the standard MSG path after receiving the end-to-end ack for all of the RMA fragments. This also avoids requiring any overall RMA state on the routers and this ensures that the completion MSG arrives after all the fragments have been completed.

b) *RMA read protocol*

The RMA read protocol is similar to the RMA write protocol in that the initiator's routing transport will manage the fragmentation of the RMA, but it will send RMA read requests. Each router will forward RMA read request to the next router until it arrives at the last router. The last router will then RMA read the data from the target to ensure the one-sided semantics of the RMA read.

Once a router has read the data, it will send a *RMA read reply* message to the next router along the path back to the initiator. When the next router has a memory buffer available, it will read the data from the previous router. When the router closest to the initiator has the fragment, it too sends a *RMA read reply* message to the initiator who then reads the data from the router. Alternately, the router could RMA write the data to the initiator and use a completion message to indicate the fragments arrival.

Completion messages will be handled as in the RMA Write case.