Local Networking

Jason Zurawski
zurawski@es.net
ESnet / Lawrence Berkeley National Laboratory

Scott Chevalier
schevali@iu.edu
Indiana University International Networks

Linux Cluster Institute (LCI) Introductory Workshop
University of Oklahoma
May 15-16, 2019

This document is a result of work by volunteer LCI instructors and is licensed under CC BY-NC-ND 4.0 (https://creativecommons.org/licenses/by-nc-nd/4.0/).
Roadmap

• Basics - TCP/IP – Firewalls – DMZs
• Technology Overview
  • Layers
  • Protocols
    • Ethernet
    • InfiniBand /OmniPath
  • Tools
• Network Design
What are We Trying to Accomplish?

• Fast?
• Latency vs Bandwidth
  • Reliable?
• Accessibility?
  • Public
  • Inside/outside
  • VPN
  • DMZ
OSI Model

- 7: Application
- 6: Presentation
- 5: Session
- 4: Transport
- 3: Network
- 2: Link
- 1: Physical
Layer 1: Physical Layer

• Transfers a stream of bits
• Defines physical characteristics
  • Connectors, pinouts
  • Cable types, voltages, modulation
  • Fibre types, lambdas
  • Transmission rate (bps)
• No knowledge of bytes or frames
Types of equipment

• Layer1: Hub, Repeater, MediaConverter
• Works at the level of individual bits
  
  ![Waveform Image]

• All data sent out of all ports
• Hence data may end up where it is not needed
Layer 2: (Data) Link Layer

- Organizes data into frames
- May detect transmission errors (corrupt frames)
- May support shared media
  - Addressing (unicast, multicast) – who should receive this frame
  - Access control, collision detection
- Usually identifies the L3 protocol carried
Example Layer 2: Ethernet

- MAC addresses
- Protocol: 2 bytes
  - e.g. 0800 = IPv4, 0806 = ARP, 86DD = IPv6
- Preamble: carrier sense, collision detection
Types of equipment (cont)

• Layer 2: Switch, Bridge
  • Receives whole layer 2 frames and selectively retransmits them
  • Learns which MAC address is on which port
  • If it knows the destination MAC address, will send it out only on that port
  • Broadcast frames must be sent out of all ports, just like a hub
  • Doesn’t look any further than L2 header
Address Learning

• After receiving a frame with the sender MAC address = X on port no Y, it “learns” that X is connected to port Y

• Learned MAC address and the corresponding port are added to the MAC Address Table

• Now, when it receives a frame with destination MAC address = X, it can send it out only on port Y, and not on other ports

• If the destination MAC address of a received frame is not in the MAC Address Table, it must be sent out on all ports (like a hub)

• If a port is connected to a single computer, then only its Ethernet address will be associated with that port

• If a port is connected to another device (hub, repeater, switch), then a number of Ethernet addresses may be associated with that port
Address Learning (cont)

- MAC addresses learned by each switch

<table>
<thead>
<tr>
<th></th>
<th>S2 MAC</th>
<th>Port</th>
<th>S3 MAC</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>1</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>2</td>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>3</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>3</td>
<td>B</td>
<td>3</td>
</tr>
</tbody>
</table>

![Diagram of network switches and connections]

- S1
  - MAC: A, B, C, D
  - Port: 1, 1, 2, 2

- S2
  - MAC: A
  - Port: 1

- S3
  - MAC: B
  - Port: 3
Layer 3: (Inter)Network Layer

• Connects Layer 2 networks together
  • Forwarding data from one network to another
  • These different networks are called subnets (short for sub-network)
• Universal datagram (Layer 3 data unit) format
• Unified addressing scheme
  • Independent of the underlying L2 network(s)
  • Addresses organized so that it can scale globally (aggregation)
• Identifies the layer 4 protocol being carried
• Fragmentation and reassembly
Example Layer 3: IPv4 Datagram

- Src, Dest: IPv4 addresses
- Protocol: 1 byte
  - e.g. 6 = TCP, 17 = UDP (see /etc/protocols)
Digging Deeper – IP (v4/v6) Addressing
IPv4 addresses

• 32-bit binary number
  • How many unique addresses in total?
    • $2^{32}$ which is 4,294,967,296 addresses

• Conventionally represented as four dotted decimal octets
Prefixes

• A range of IP addresses is given as a prefix, e.g. 192.0.2.128/27

Prefix length /27 -> First 27 bits are fixed

Lowest address:

192 . 0 . 2 . 128
110000000000000000010000000

Highest address:

192 . 0 . 2 . 159
11000000000000000001001111
IPv4 “Golden Rules”

1. All hosts on the same L2 network must share the same prefix
2. All hosts with the same prefix have different host part
3. Host part of all-zeros and all-ones are reserved

• Golden Rules for 192.0.2.128/27
  • Lowest 192.0.2.128 = network address
  • Highest 192.0.2.159 = broadcast address
  • Usable: 192.0.2.129 to 192.0.2.158
  • Number of usable addresses: 32 - 2 = 30
IPv6 addresses

• 128-bit binary number
  • How many unique addresses in total?
  • $3.402823669209 	imes 10^{38}$

• Conventionally represented in hexadecimal – 8 words of 16bits
  
  **2607:8400:2880:0004:0000:0000:80DF:9D13**

• Leading zeros can be dropped

• The right-most contiguous run of all-zero words can be replaced by "::"

  **2607:8400:2880:4::80DF:9D13**
# Hexadecimal

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>1100</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>1101</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>1110</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>1111</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

\[
0000 = 00000000000000000000000000000000
FFFF = 11111111111111111111111111111111
\]
IPv6 rules

• With IPv6, every subnet is /64 *

• The remaining 64 bits can be assigned by hand, or picked automatically
  • all-zeros address is reserved * - Subnet-Router Anycast address

• There are special prefixes
  • e.g. link-local addresses start with FE80::

• Total available IPv6 space is ≈ 261 subnets

* Except /127 recommended for point-to-point links (RFC 6164), in which case the all-zeros address is allowed
IPv6 addressing

• Typical end-user allocation is /48
Types of equipment (cont)

• Layer 3: Router

• Looks at the destination IP in its Forwarding Table to decide where to send next

• Collection of routers managed together is called an “Autonomous System”

• The forwarding table can be built by hand (static routes) or dynamically
  • Within an AS: IGP (e.g. OSPF, IS-IS)
  • Between ASes: EGP (e.g. BGP)
Traffic Domains

- **Broadcast Domain**: all devices on the same sub-network.

- **Collision Domain**: where several devices share one communication medium (for example, on wireless networks).
What is a Router?

• A router is a layer 3 device
• Used for interconnecting networks at layer 3
• A router generally has at least two interfaces
  • With VLANs a router can have only one interface (known as “router on a stick”)
• A router looks at the destination address in the IP packet, and decides how to forward it
The Routing Table

• Each router/host has a routing table, indicating the path or the next hop for a given destination host or a network
• The router/host tries to match the destination address of a packet against entries in the routing table
• If there is a match, the router forwards it to the corresponding gateway router or directly to the destination host
• Default route is taken if no other entry matches the destination address
The Routing Table

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next-Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.40.0.0/16</td>
<td>192.248.40.60</td>
<td>Ethernet0</td>
</tr>
<tr>
<td>192.248.0.140/30</td>
<td>Directly connected</td>
<td>Serial1</td>
</tr>
<tr>
<td>192.248.40.0/26</td>
<td>Directly connected</td>
<td>Ethernet0</td>
</tr>
<tr>
<td>192.248.0.0/17</td>
<td>192.248.0.141</td>
<td>Serial1</td>
</tr>
<tr>
<td>203.94.73.202/32</td>
<td>192.248.40.3</td>
<td>Ethernet0</td>
</tr>
<tr>
<td>203.115.6.132/30</td>
<td>Directly connected</td>
<td>Serial0</td>
</tr>
<tr>
<td>Default</td>
<td>203.115.6.133</td>
<td>Serial0</td>
</tr>
</tbody>
</table>

Typical routing table on a simple edge router
IP Routing – finding the path

• Routing table entry (the path) is created by the administrator (static) or received from a routing protocol (dynamic)

• More than one routing protocol may run on a router
  • Each routing protocol builds its own routing table (Local RIB)

• Several alternative paths may exist
  • Best path selected for the router’s Global routing table (RIB)

• Decisions are updated periodically or as topology changes (event driven)

• Decisions are based on:
  • Topology, policies and metrics (hop count, filtering, delay, bandwidth, etc.)
IP route lookup

• Based on destination IP address

• “longest match” routing
  • More specific prefix preferred over less specific prefix

• Example:
  • packet with destination of 2001:DB8:1::1/128 is sent to the router announcing 2001:DB8:1::/48 rather than the router announcing 2001:DB8::/32.
Layer 4: Transport Layer

• Identifies the endpoint process
• Another level of addressing (port number)
• May provide reliable delivery
  • Streams of unlimited size
  • Error correction and retransmission
  • In-sequence delivery
  • Flow control
• Might just be unreliable datagram transport
Example Layer 4: UDP

- Port numbers: 2 bytes
  - Well-known ports: e.g. 53 = DNS
  - Ephemeral ports: ≥1024, chosen dynamically by client
Layers 5, 6, 7

• #5: Session Layer: long-lived sessions
  • Re-establish transport connection if it fails
  • Multiplex data across multiple transport connections

• #6: Presentation Layer: data reformatting
  • Character set translation

• #7: Application Layer: the actual work you want to do
  • Protocols specific to each application

• These do not exist in the TCP/IP suite: the application is responsible for these functions
OSI vs TCP/IP

Source: William Stallings
“Data and Computer Communications”
Encapsulation

- Each layer provides services to the layer above
- Each layer makes use of the layer below
- Data from one layer is encapsulated in frames of the layer below

**Ex**
- L4 segment contains part of stream of application protocol
- L3 datagram contains L4 segment
- L2 frame has L3 datagram in data portion
And Now for Something Completely Different – Other Interconnection Strategies
High Speed Interconnect

• Significant percentage of the cost of a cluster (1/3 or more)
• High bandwidth, low latency
• Exact requirements depend on cluster workload
• Typically, want some hardware offload capability
• RDMA
• Needs to present a software interface compatible with your desired middleware (MPI?)
• Reliability desirable (may carry filesystem and other important traffic)
• Typically short range (single room, or at least single building)
• Proprietary and open solutions available
Key Differences from TCP/IP

• InfiniBand (etc) are often compared to traditional network approaches such as TCP/IP/Ethernet.

• InfiniBand is based on “layers” but there are drastic differences in boundaries between the layers.
  • Key difference: messaging service that applications can access directly.

• Messages are used for
  • Storage
  • InterProcess Communication (IPC)
  • anything that requires an application to communicate with others in its environment.

• The application has more control over this protocol, vs. the byte-stream/sockets in TCP.
Application/Network interaction
Hardware Offload - RDMA

• “Remote Direct Memory Access” – The CPU designates a memory region, which the network cards then transfer

• CPU can (and should!) continue doing computation during the transfer

• Exposed to user software through various special purpose APIs

• LibFabric is an up-and-coming alternative

• One API for a variety of hardware (at least in theory)

• Some Vendors have their own – PSM, Portals, etc

• Software often wrapped with a higher level middleware layer, almost

• The Open Fabric Alliance conference/mailing lists are a good source of information about RDMA

• always MPI for HPC use cases
Hardware Offload – RDMA – High level workflow

- Create a TCP/IP connection to the nodes of interest and exchange their high-speed interconnect addresses (IB and similar generally don’t have a DNS like thing)
- Pin a memory region for transfer on both nodes (so the kernel won’t move the data before the network card is done)
- Open a context to the network card and create a queue pair
- Exchange metadata about the transfer over TCP
- Put a work request in the queue specifying the region to transfer
- Wait for a completion message (or better, do something else with the CPU).
- Cleanup
RDMA Configuration Pitfall: Pinned Memory

• Memory must be “pinned” for RDMA transfers so that the NIC knows where to find (or put) the data being transferred is in physical memory and can be sure that the OS won’t move it.

• Having a lot of pinned memory can interfere with normal OS operations like paging and the handling of out-of-memory events, so the amount a user can pin is usually restricted.

• If a user can’t pin their entire buffer, the transfer will fail (unless they are using very smart middleware).

• This is implemented on Linux with “ulimit”.

• `printf "%s\n%s" "soft memlock unlimited" "hard memlock unlimited" >> /etc/security/limits.conf`
Specific Interconnect Hardware - Options

• InfiniBand
• OmniPath
• Cray Proprietary (Aries at the moment)
• SGI Proprietary (NUMALINK)
• Historically:
  • IBM Proprietary – BlueGene
  • Myrinet
InfiniBand

- Popular since the early 00’s
- “Lossless”
- Standard, but only a few current manufactures
  - Nearly everyone (with IB) in HPC uses Mellanox gear (or someone else’s stuff made with Mellanox ASICs)
  - Oracle and Obsidian also have some IB hardware
  - QLogic’s IB was bought by Intel in 2012
- Speeds are rated by generation and number of lanes
  - Current generation is HDR: 50 gigabits/lane
  - Current implementations almost always use 4x – effectively 200 gigabits
  - EDR (25gb/lane) still common, as are FDR (14gb/lane) and QDR (8gb/lane)
- Modern implementations use the QSFP+ connector (fiber or short copper runs)
Block Diagram

- Application
- InfiniBand Messaging Service
- S/W Transport Interface
- Transport
- Network
- Link
- Physical

RDMA Message Transport Service
InfiniBand

- Spanning-tree-like capabilities are built-in (via the SM)
- Generally the switch side “Just Works”TM
- Auto-speed changes
- Client-side gets trickier, especially for performance tuning
  - OFED (Open Fabrics Enterprise Distribution)
  - Tweaking kernel parameters
- Do yourself a favor – don’t mix vendors. It theoretically works but...
- I suggest also keeping firmware versions consistent
- You can mix speeds (with caveats)
InfiniBand

• Centrally Managed (everyone uses OpenSM)
• Verbs RDMA is the native API
• IP (or even emulated Ethernet with recent kernels) available with significantly reduced performance
• Statically routed via a per-port linear forwarding table in nearly every case
• Subnet routing barely supported (hardware that can do this is very new)
• 16bit assigned addresses
• 64bit hardware addresses
• 128bit addresses for inter-subnet routing (again, barely any real hardware)
InfiniBand - Terms

• LID: Local assigned address (like an IP address)
• GUID: Hardware address (like an ethernet MAC address)
• SM: Subnet manager: Centralized software that determines the routing table and various other network properties
• Queue Pair: A pair of queues used for RDMA transactions (one send queue, one receive queue). Could be a software construct, but they usually require some hardware resources on the NIC and are therefore finite in number
• HCA: “Host Channel Adaptor” : A network card (NIC)
• Other high-speed interconnects sometimes call these “HFI”: Host Fabric Interfaces or HBA: Host Bus Adaptors
OmniPath

- Shiny, new competitor to InfiniBand from Intel
  - Result of Intel’s purchase of QLogic tech. and Cray IP
  - Hardware incompatible with IB
- Software compatible with IB
- PCIe cards don’t have as much hardware offloading as IB
- Intel chips with OPA on-board Real Soon NowTM
- Potential to be less expensive
- In active development
Proprietary Interconnects
• Can be good or bad – Single vendor is usually a good idea
• Consider whether your workload works on the vendor’s MPI
• Does your PFS support the vendor interconnect?
• Expandability?
• Can you interoperate with other technologies? Do you want to?
• Can the vendor support the product for your desired system life?
• Smaller support options
  • Lots of InfiniBand installs – you can Google
  • ...but some vendors have very good user groups.
Proprietary Interconnects – NUMALink/UV

• Maybe a cluster/traditional network isn’t right for you
• Multi-socket machines have a “network” on the motherboard
• Interconnecting the CPUs
• NUMALink lets you extend this to a few racks of machines while maintaining a single system image
• Run threaded applications (or even single processes, really slowly) with (up to) the resources of several racks of hardware.
• Expensive, but really convenient for non-distributed memory workloads
• NUMALink is made by SGI/HPE, but others have tried similar things (Oracle still has some 16 socket SPARC systems for example).
Ethernet vs InfiniBand (IB) and OmniPath (OPA)

• Most clusters have both Ethernet and IB or OPA
  • Advantages and disadvantages of each
  • (Relatively) slow Ethernet for external access, file copying, etc.
  • IB and OPA for MPI

• RDMA vs IP

• Management (PXE, Lights-Out, IPMI, infrastructure)
Management Networks

• Two types:
  • ssh/software updates/config management/etc. (user accessible)
  • Admin only

• Uses:
  • Monitoring/sensors
  • Depending on compute node type (stateless) maybe be used during boot
  • Typically used for Lights-Out Hardware interface
  • IPMI
  • Might want to put IPMI/hardware on separate VLAN(s) for security
  • Needs to be cheap and reliable but not necessarily fast
  • Sometimes used for compute node internet access, especially if your main interconnect doesn’t support IP well (more on that later)
  • Usually gigabit Ethernet
Putting it Together – Network Design
Campus Network Rules

• Minimize number of network devices in any path
• Use the hub and spoke (star) configuration design pattern
• Segment your network with routers at the core/middle
• Provide services near the core
• Think carefully about where to firewall and where to NAT
Minimize Number of Network Devices in the Path

• Build hub and spoke (sometimes called star) networks

• Not daisy chained (sometimes called cascaded) networks
Hub and Spoke at Campus Level

• At the campus level, best practices are to build hub and spoke networks
• The hub at the campus level is often called the core
• Best practices are to route at the core
  • This segments the network into independent subnets
  • Limits broadcasts
Basic Idea
Advanced Idea
Network Topology for a cluster

• Design Goals:
  • Maximum bandwidth between any two points
  • Minimum latency between any two points
  • Minimize hops – most of the latency is in switching/processing
  • Keep the cables short
    • ... Or use less of them
  • Costs are per {switch, port, cable}
  • Want collective operations to be fast (for most HPC workloads)
  • Redundancy, filesystems, etc may impose additional requirements
Internal/External Interfaces to the Cluster

• Who has access?
• Who provides access?
• What internal resources can they see?
• VPN considerations
• DMZ considerations
Requirements

• High-Speed Interconnect
  • Low Latency
  • High Bandwidth
  • Reliable
  • User software support (MPI, etc)
Requirements (cont)

• Internet/External Access
  • Bandwidth
  • Access to/from external sites
  • Security
  • Reliable
Requirements (cont)

• Management
  • Cheap
  • Reliable Hardware/Software Support
  • PXE
  • IPMI
  • Secure (Isolation from users)
Network Topology for a cluster

• Let’s start the discussion by physical and logical topologies

• I’ll give some examples and you tell me the advantages and disadvantages of each
  • (Hint: If there were no disadvantages to any of these, we would only have one slide which showed what everybody does)
Ex
Ex
Ex
Internet / WAN

• Usually Ethernet
• Usually have a security boundary between the WAN and the rest of the cluster
• Dedicated login/head nodes
• External interface is typically ssh
• May have other site specific firewall considerations
• Compute nodes are typically behind some kind of NAT, but could be publicly addressable (if you have address space to burn).
• Consider redundancy and whether your applications can handle multipath routing
• Lots of vendors/site specific – hard to give general advice
External Connectivity
External Connectivity + Security

- ISP
- Border Router
- Core Router
- All router interfaces on a separate subnet
- Servers with sensitive data
- Fiber optic links to remote buildings
- Student servers
- Different subnets with different firewall rules – block students from accessing sensitive data
Big Picture
Local Diagnostic Tools
Diagnostics - Hardware

• Optics seated?
  • Most lock in some way
• Fibers crossed?
  • Try flipping one side. LC connectors may be keyed incorrectly
• Cables pinched/bent?
  • Check the bend radius specification
• Slow performance? Try reseating your NIC and/or SFP
• Ensure ports are enabled on both sides
• ifconfig [interface] up
• InfiniBand: ensure the SM is running, otherwise you may not even get a link light
Diagnostics

• ethtool [interface]
• ibstat
• ibv_devinfo

```bash
# ibstat
CA 'mlx5_0'

CA type: MT4115
Number of ports: 1
Firmware version: 12.14.2036
    Hardware version: 0
Node GUID: 0x7cfe9003008ddacc
    System image GUID: 0x7cfe9003008ddacc
Port 1:
    State: Active
    Physical state: LinkUp
    Rate: 100
    Base lid: 5
    LMC: 0
    SM lid: 29
    Capability mask: 0x2651e848
    Port GUID: 0x7cfe9003008ddacc
    Link layer: InfiniBand
```
Diagnostics - InfiniBand

- `Ibdiagnet`
  - General diagnostics
- `Ibnetdiscover`
  - What can we talk to?
- `Ibtopodiff`
  - Are the cables in the right places?
- `Ibportstate`
  - Is the port enabled?
- `perfquery`
  - Each port has counters for various events. The vendor can tell you which are problematic in what quantity
Diagnostics - RDMA

• ib_write_bw #server

• ib_write_bw -b --run_infinity ip_address_of_server #client
Diagnostics – Packet Capture

• Extremely helpful for figuring out what’s happening to your packets
• Ethernet: tcpdump
• IB (Mellanox Only): ibdump
• Visualization: wireshark
Local Networking

Jason Zurawski  
zurawski@es.net  
ESnet / Lawrence Berkeley National Laboratory

Scott Chevalier  
schevali@iu.edu  
Indiana University International Networks

Linux Cluster Institute (LCI) Introductory Workshop  
University of Oklahoma  
May 15-16, 2019