Network Performance - Theory

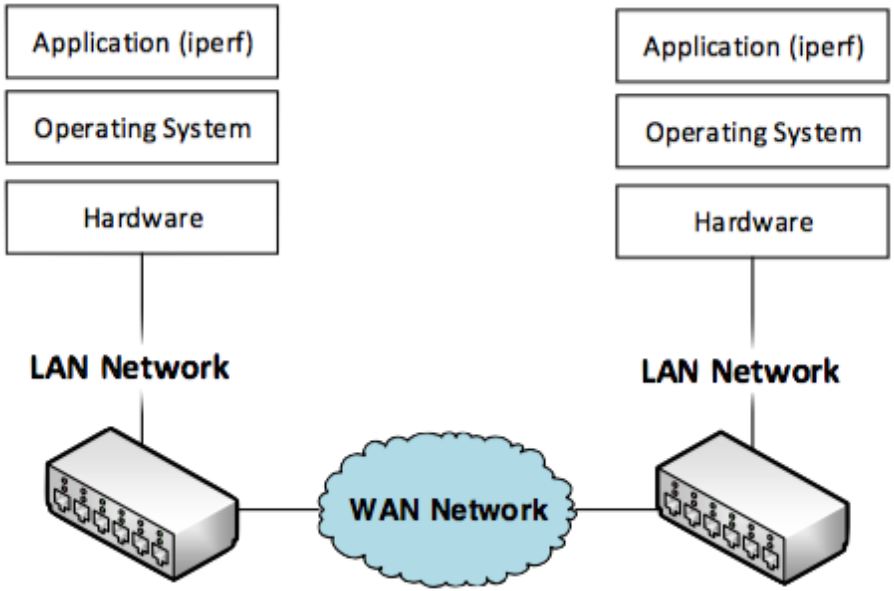
Event

Presenter, Organization, Email
Date

Introduction

- perfSONAR is a tool to measure “end-to-end” network performance. What does this imply:
  - End-to-end: the entire network path between perfSONAR Hosts
    - Applications Software
    - Operating System
    - Host Hardware
    - @ Each Hop: transition between OSI Layers in routing/switching devices (e.g. Transport to Network Layer, etc.), buffering, processing speed
    - Flow through security devices
  - No easy way to separate out the individual components by default – the number the tool gives has them all ***combined***

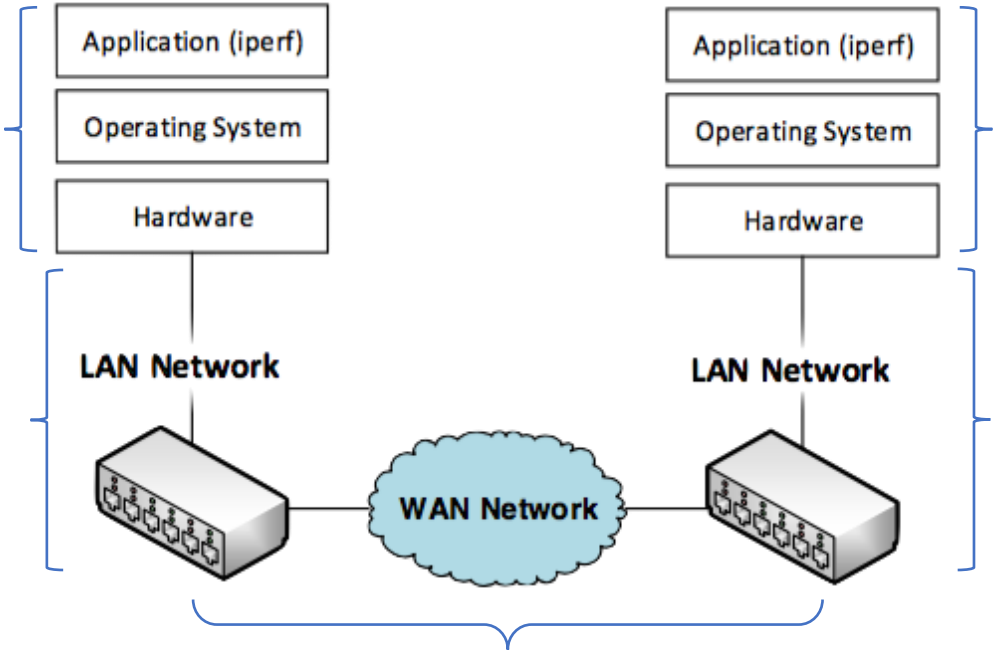
Initial End-to-End Network



Initial End-to-End Network

Src Host Delay:
- Application writing to OS (kernel)
- Kernel writing via memory to hardware
- NIC writing to network

Src LAN:
- Buffering on ingress interface queues
- Processing data for destination interface
- Egress interface queuing
- Transmission/Serialization to wire



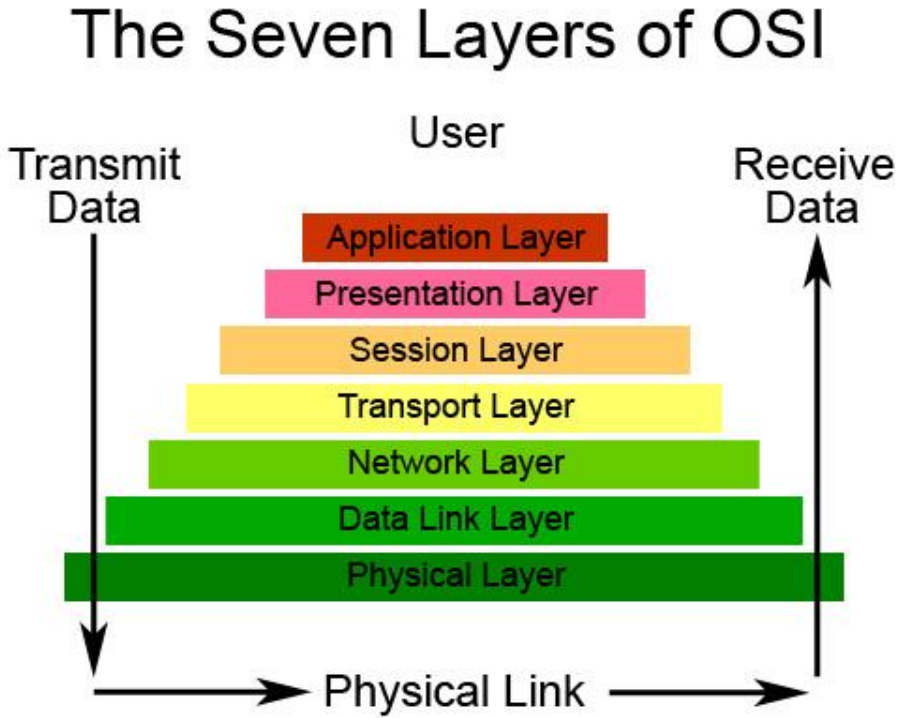
Dst Host Delay:
- NIC receiving data
- Kernel allocating space, sending to application
- Application reading/acting on received data

Dst LAN:
- Buffering on ingress interface queues
- Processing data for destination interface
- Egress interface queuing
- Transmission/Serialization to wire

WAN:
- Propagation delay for long spans
- Ingress queuing/processing/egress queuing/serialization for each hop

ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN

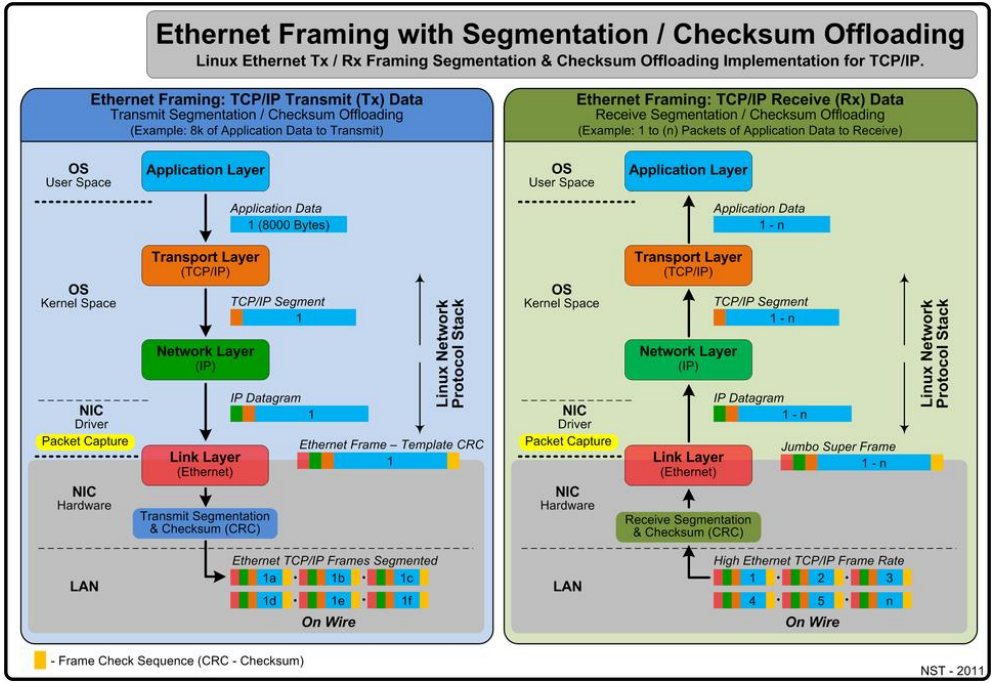
OSI Stack Reminder



- The demarcation between each layer has an API (e.g. the 'narrow waist' of an hourglass)
- Some layers are more well defined than others:
  - Within an application the job of presentation and session may be handled
  - The operating system handles TCP and IP, although these are separate libraries
  - Network/Data Link occur within network devices (Routers, Switches)

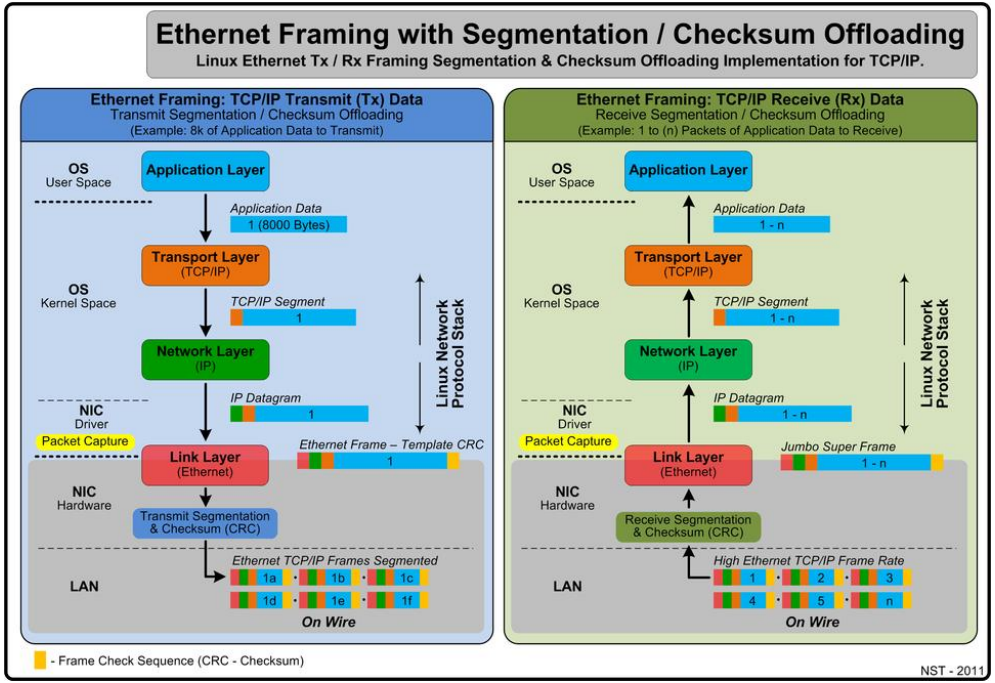
Host Breakout

- Most applications are written in ‘user space’, e.g. special section of the OS that is jailed from ‘kernel space’.
- Requests to use functions like the network are done by using system calls through an API (e.g. open a ***socket*** so I can communicate with a remote host)
- The TCP/IP libraries are within the kernel, they receive the request and take care of the heavy lifting of converting the data from the application (e.g. a large chunk of memory) into individual packets for the network
- The NIC will then encapsulate into Link layer protocol (e.g. ethernet frames) and send onto the wire for the next hop to deal with



Host Breakout

- The receive side works similar – just in reverse
- Frames come off of the network and into the NIC. The onboard processor will extract packets, and pass them to the kernel
- The kernel will map the packets to the application that should be dealing with them
- The application will receive the data via the API
- Note – the TCP/IP libraries manage things like data control. The application only sees a socket, and knows that it will send in data, and it will make it to the other side. It is the job of the library to ensure reliable delivery



Network Device Breakout

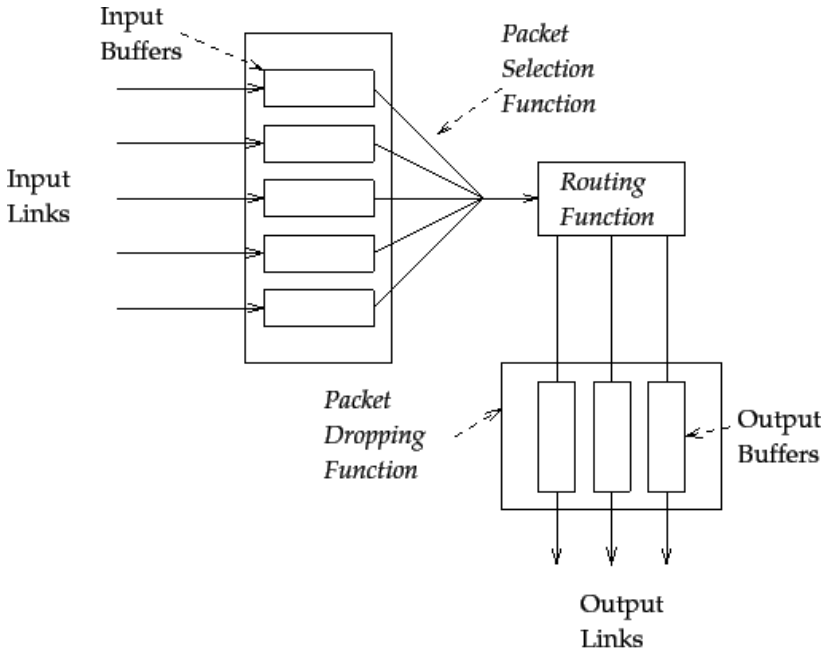


Figure 1: Basic Router Architecture

Network Device Breakout

- Data arrives from multiple sources

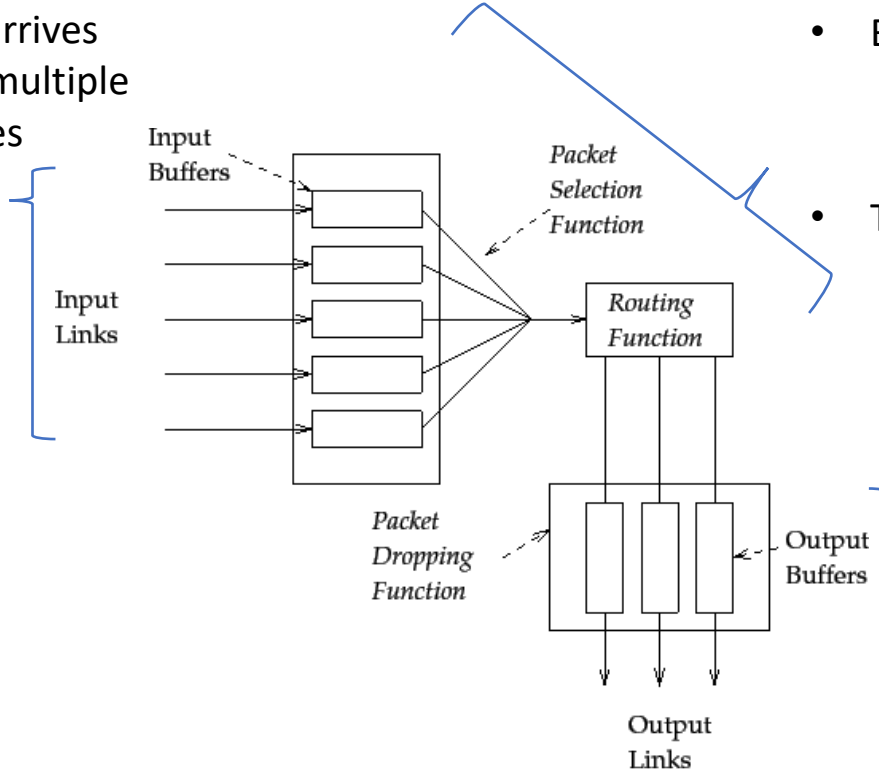
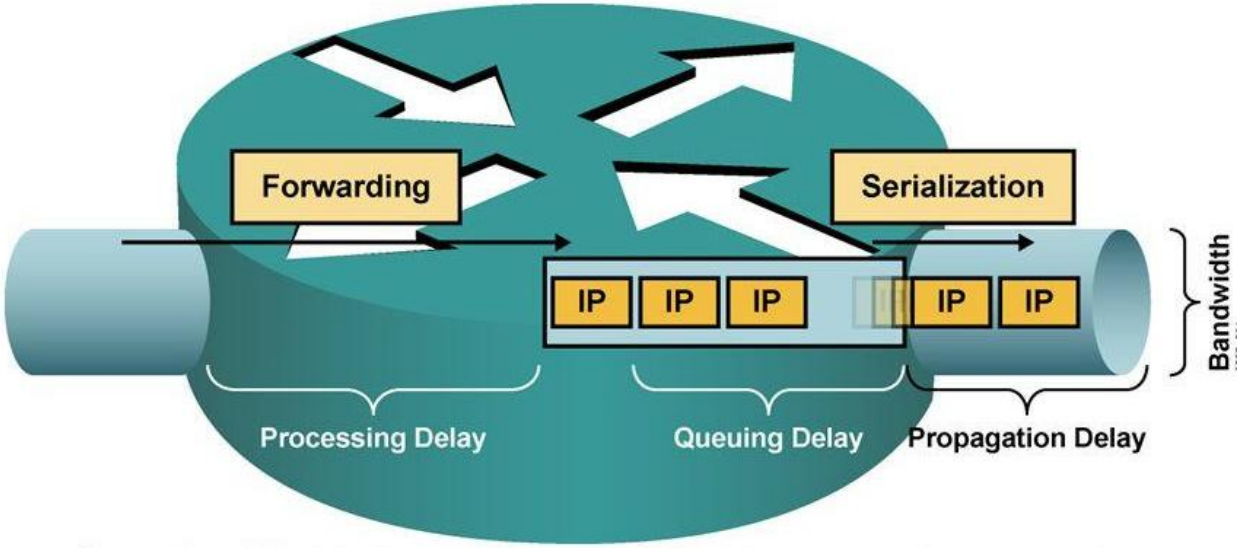


Figure 1: Basic Router Architecture

- Buffers have a finite amount of memory
  - Some have this per interface
  - Others may have access to a shared memory region with other interfaces
- The processing engine will:
  - Extract each packet/frame from the queues
  - Pull off header information to see where the destination should be
  - Move the packet/frame to the correct output queue
- Additional delay is possible as the queues physically write the packet to the transport medium (e.g. optical interface, copper interface)

Network Device Breakout – Delays

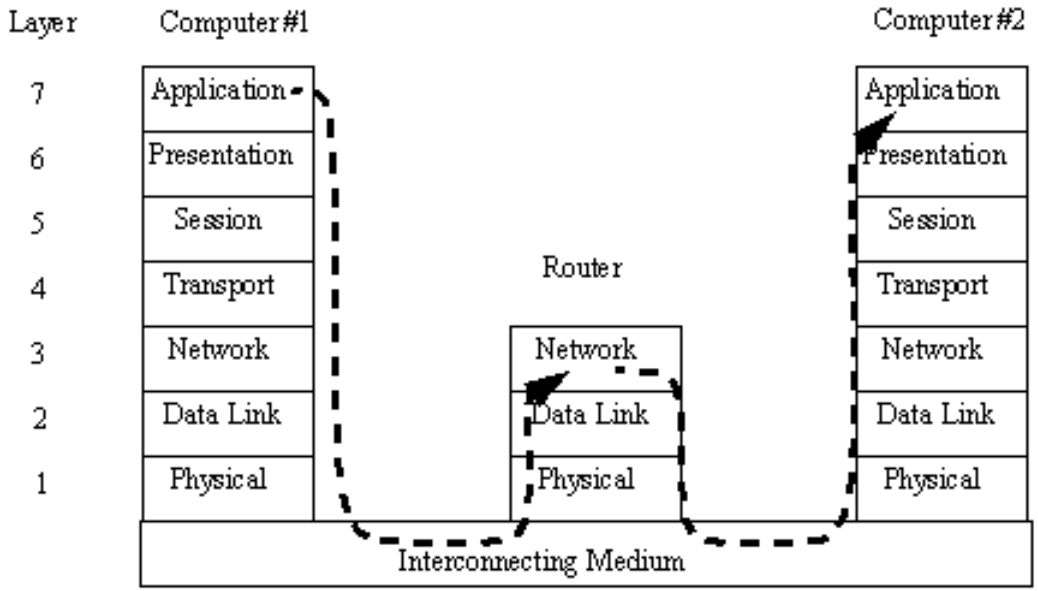


- **Processing delay:** The time it takes for a router to take the packet from an input interface, examine it, and put it into the output queue of the output interface.
- **Queuing delay:** The time a packet resides in the output queue of a router.
- **Serialization delay:** The time it takes to place the "bits on the wire."
- **Propagation delay:** The time it takes for the packet to cross the link from one end to the other.

ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN

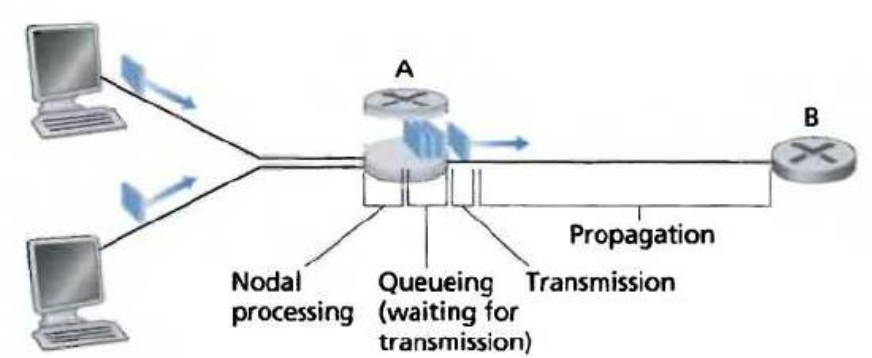
Network Devices & OSI

- Not every device will care about every layer
- Hosts understand them all via various libraries
- Network devices only know up to a point:
  - 'Routers' know up to the 'Network Layer'. They will make the choice of sending to the next hop based on Network Layer headers. E.g. TCP information – IP addresses
  - 'Switches' know up to the 'Link Layer'. They will make the choice of sending to the next hop based on Link Layer headers. E.g. MAC addressing from the IP
- Each hop has the hardware/software to pull of the encapsulated data to find what it needs



"End-to-End"

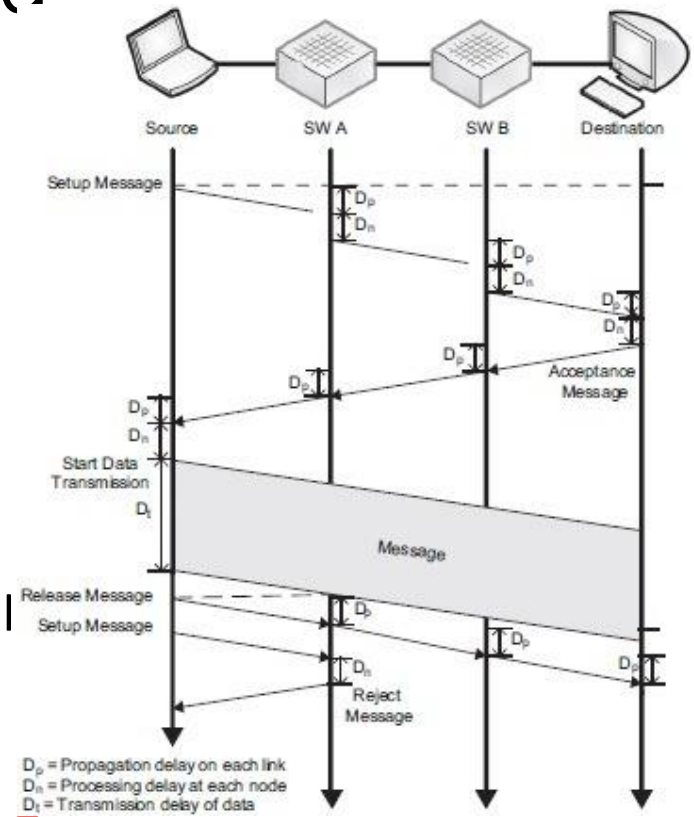
- A network user interfaces with the network via a tool (data movement application, portal). They only get a single piece of feedback – how long the interaction takes
- In reality – it's a complex series of moves to get the data end to end, with limited visibility by default
  - Delays on the source host
  - Delays in the source LAN
  - Delays in the WAN
  - Delays in the destination LAN
  - Delays on the destination host



The nodal delay at router A

"End-to-End"

- The only way to get visibility is to rely on instrumentation at the various layers:
  - Host level monitoring of key components (CPU, memory, network)
  - LAN/WAN level monitoring of individual network devices (utilization, drops/discards, errors)
  - End-to-end simulations
- The later one is tricky – we want to 'simulate' what a user would see by having our own (well tuned) application tell us how it can do across the ***common network substrate***



Dereferencing Individual Components

- Host Performance
  - Software Tools – Ganglia, Host SNMP + Cacti/MRTG
- Network Device Performance
  - SNMP/TL1 Passive Polling (e.g. interface counters, internal behavior)
- Software Performance
  - ??? This depends heavily on how well the software (e.g. operating system, application) is instrumented.
- End-to-end
  - perfSONAR active tools (iperf3, owamp, etc.)

Takeaways

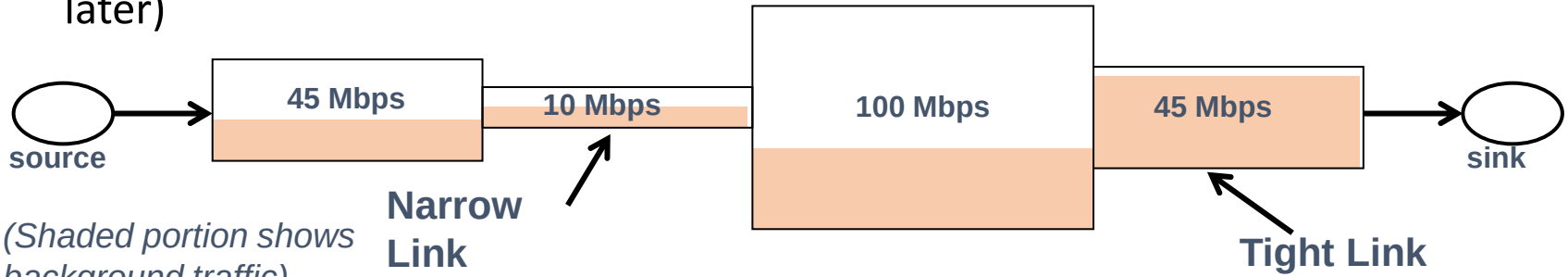
- Since we want “network” performance – we want to remove the host hardware/operating system/applications from the equation as much as possible
- Things that we can do on our own, or that we get for free by using perfSONAR:
  - Host Hardware: Choosing hardware matters. There needs to be predictable interactions between system components (NIC, motherboard, memory, processors)
  - Operating System: perfSONAR features a tuned version of CentOS. This version eliminates extra software and has been modified to allow for high performance networking
  - Applications: perfSONAR applications are designed to make minimal system calls, and do not involve the disk subsystem. The performance they report is designed to be as low-impact on the host to achieve realistic network performance

Lets Talk about IPERF

- Start with a definition:
  - ***network throughput*** is the rate of successful message delivery over a communication channel
  - Easier terms: how much data can I shovel into the network for some given amount of time
  - Things it includes: the operating system, the host hardware, and the entire network path
- What does this tell us?
  - Opposite of utilization (e.g. its how much we can get at a given point in time, minus what is utilized)
  - Utilization and throughput added together are “capacity”
- Tools that measure throughput are a simulation of a real work use case (e.g. how well could bulk data movement perform)

What IPERF Tells Us

- Lets start by describing throughput, which is vague.
  - Capacity: link speed
    - Narrow Link: link with the lowest capacity along a path
    - Capacity of the end-to-end path = capacity of the narrow link
  - Utilized bandwidth: current traffic load
  - Available bandwidth: capacity – utilized bandwidth
    - Tight Link: link with the least available bandwidth in a path
  - Achievable bandwidth: includes protocol and host issues (e.g. BDP!)
- All of this is “memory to memory”, e.g. we are not involving a spinning disk (more later)



BWCTL Example (iperf3)

```
[zurawski@wash-pt1 ~]$ bwctl -T iperf3 -f m -t 10 -i 2 -c sunn-pt1.es.net
bwctl: run_tool: tester: iperf3
bwctl: run_tool: receiver: 198.129.254.58
bwctl: run_tool: sender: 198.124.238.34
bwctl: start_tool: 3598657653.219168
Test initialized
Running client
Connecting to host 198.129.254.58, port 5001
[ 17] local 198.124.238.34 port 34277 connected to 198.129.254.58 port 5001
[ ID] Interval           Transfer     Bandwidth       Retransmits
[ 17]   0.00-2.00   sec   430 MBytes  1.80 Gbits/sec  2
[ 17]   2.00-4.00   sec   680 MBytes  2.85 Gbits/sec  0
[ 17]   4.00-6.00   sec   669 MBytes  2.80 Gbits/sec  0
[ 17]   6.00-8.00   sec   670 MBytes  2.81 Gbits/sec  0
[ 17]   8.00-10.00  sec   680 MBytes  2.85 Gbits/sec  0
[ ID] Interval           Transfer     Bandwidth       Retransmits
    Sent
[ 17]   0.00-10.00  sec  3.06 GBytes  2.62 Gbits/sec  2
    Received
[ 17]   0.00-10.00  sec  3.06 GBytes  2.63 Gbits/sec

iperf Done.
bwctl: stop_tool: 3598657664.995604

SENDER END
```

N.B. This is what perfSONAR Graphs – the average of the complete test

Summary

- We have established that our tools are designed to measure the network
- For better or for worse the ‘network’ is also our host hardware, operating system, and application
- To get the most accurate measurement, we need:
  - Hardware that performs well
  - Operating systems that perform well
  - Applications that perform well

Network Performance - Theory

Event

Presenter, Organization, Email
Date