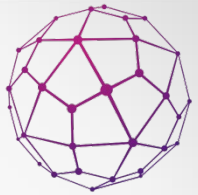


APPLIED
RESEARCH
CENTER FOR
COMPUTER
NETWORKS

Computer Network Modelling #1

PhD. Antonenko V.A.

APPLIED RESEARCH CENTER FOR COMPUTER NETWORKS

Goals Of This Lecture

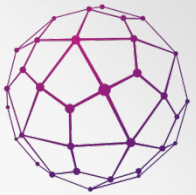
- Introduce Modeling
- Introduce Simulation
- What We Need for Simulation?
- Simulation Model Building
- Mininet

- “Learn by Doing”--Lots of Case Studies

What Is A Model ?

A Representation of an object, a system, or an idea in some form other than that of the entity itself.

(Shannon)



APPLIED RESEARCH CENTER FOR COMPUTER NETWORKS

Types of Models:

Physical

(Scale models, prototype plants, …)

Mathematical

(Analytical queueing models, linear programs, simulation)

What is Simulation?

- A Simulation of a system is the operation of a model, which is a representation of that system.
- The model is amenable to manipulation which would be impossible, too expensive, or too impractical to perform on the system which it portrays.
- The operation of the model can be studied, and, from this, properties concerning the behavior of the actual system can be inferred.

Applications:

- Designing and analyzing manufacturing systems
- Evaluating H/W and S/W requirements for a computer system
- Evaluating a new military weapons system or tactics
- Determining ordering policies for an inventory system
- Designing communications systems and message protocols for them

Applications:(continued)

- Designing and operating transportation facilities such as freeways, airports, subways, or ports
- Evaluating designs for service organizations such as hospitals, post offices, or fast-food restaurants
- Analyzing financial or economic systems

Steps In Simulation and Model Building

1. Define an achievable goal
2. Involve the end-user
3. Choose the appropriate simulation tools
4. Model the appropriate level(s) of detail
5. Start early to collect the necessary input data

Steps In Simulation and Model Building(cont'd)

6. Provide adequate and on-going documentation
7. Develop a plan for adequate model verification

   (Did we get the "right answers ?")
8. Develop a plan for model validation

   (Did we ask the "right questions ?")
9. Develop a plan for statistical

Define An Achievable Goal

"To model the…" is NOT a goal!

"To model the…in order to select/ determine feasibility/…is a goal.

Goal selection is not cast in concrete

Goals change with increasing insight

Put together a complete mix of skills on the team

We Need:

-Knowledge of the system under investigation

-System analyst skills (model formulation)

-Model building skills (model Programming)

-Data collection skills

-Statistical skills (input data representation)

Put together a complete mix of skills on the team(continued)

We Need:

-More statistical skills (output data analysis)

-Even more statistical skills (design of experiments)

-Management skills (to get everyone pulling in the same direction)

INVOLVE THE END USER

-Modeling is a selling job!

-Does anyone believe the results?

-Will anyone put the results into action?

-The End-user (your customer) can (and must) do all of the above BUT, first he must be convinced!

-He must believe it is HIS Model!

Choose The Appropriate Simulation Tools

Assuming Simulation is the appropriate means, three alternatives exist:

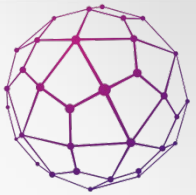
1. Build Model in a General Purpose Language
2. Build Model in a General Simulation Language
3. Use a Special Purpose

MODELLING GENERAL PURPOSE LANGUAGES

APPLIED RESEARCH CENTER FOR COMPUTER NETWORKS

- Advantages:
  - Little or no additional software cost
  - Universally available (portable)
  - No additional training (Everybody knows…(language X) !)
- Disadvantages:
  - Every model starts from scratch
  - Very little reusable code
  - Long development cycle for each model
  - Difficult verification phase

MODELING W/ GENERAL SIMULATION LANGUAGES



- Advantages:
  - Standardized features often needed in modeling
  - Shorter development cycle for each model
  - Much assistance in model verification
  - Very readable code
- Disadvantages:
  - Higher software cost (up-front)
  - Additional training required
  - Limited portability

MODELING W/ SPECIAL-PURPOSE SIMUL. PACKAGES

- Advantages
  - Very quick development of complex models
  - Short learning cycle
  - No programming--minimal errors in usage
- Disadvantages
  - High cost of software
  - Limited scope of applicability
  - Limited flexibility (may not fit your specific application)

SPECIAL PURPOSE PACKAGES USED FOR SIMUL.

- NETWORK II.5
  - Simulator for computer systems
- OPNET
  - Simulator for communication networks, including wireless networks
- COMNET III
  - Simulator for communications networks
- SIMFACTORY
  - Simulator for manufacturing operations

THE REAL COST OF SIMULATION

Many people think of the cost of a simulation only in terms of the software package price.

There are actually at least three components to the cost of simulation:

1. Purchase price of the software
2. Programmer / Analyst time
3. “Timeliness of Results”

TERMINOLOGY

- System
  - A group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose.
  - Entity
  - An object of interest in the system.
  - E.g., customers at a bank

TERMINOLOGY (continued)

- Attribute
  - a property of an entity
  - E.g., checking account balance
- Activity
  - Represents a time period of specified length.
  - Collection of operations that transform the state of an entity
  - E.g., making bank deposits

TERMINOLOGY (continued)

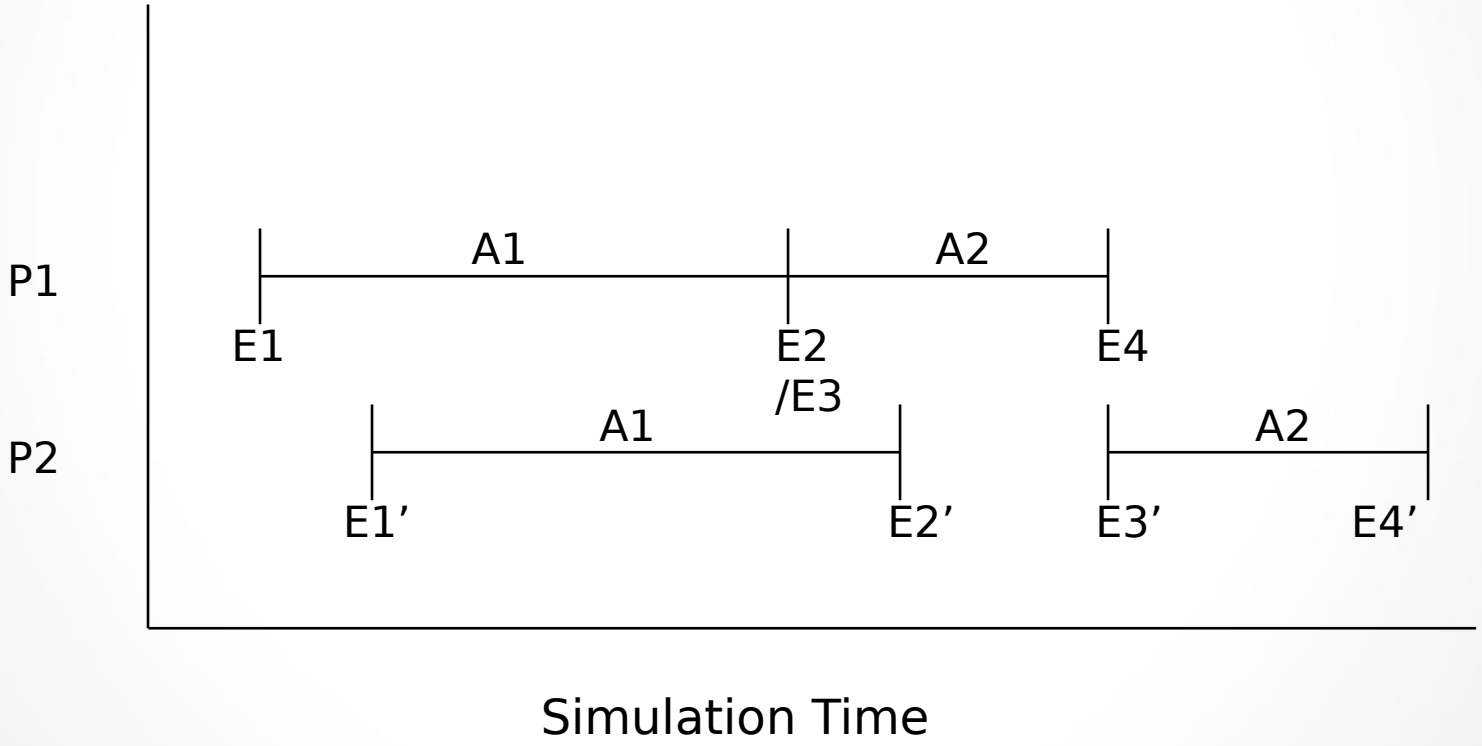
- Event:
  - change in the system state.
  - E.g., arrival; beginning of a new execution; departure
- State Variables
  - Define the state of the system
  - Can restart simulation from state variables
  - E.g., length of the job queue.

TERMINOLOGY (continued)

- Process
  - Sequence of events ordered on time

- ✷ Note:
  - the three concepts(event, process,and activity) give rise to three alternative ways of building discrete simulation models

A GRAPHIC COMPARISON OF DISCRETE SIMUL. METHODOLOGIES



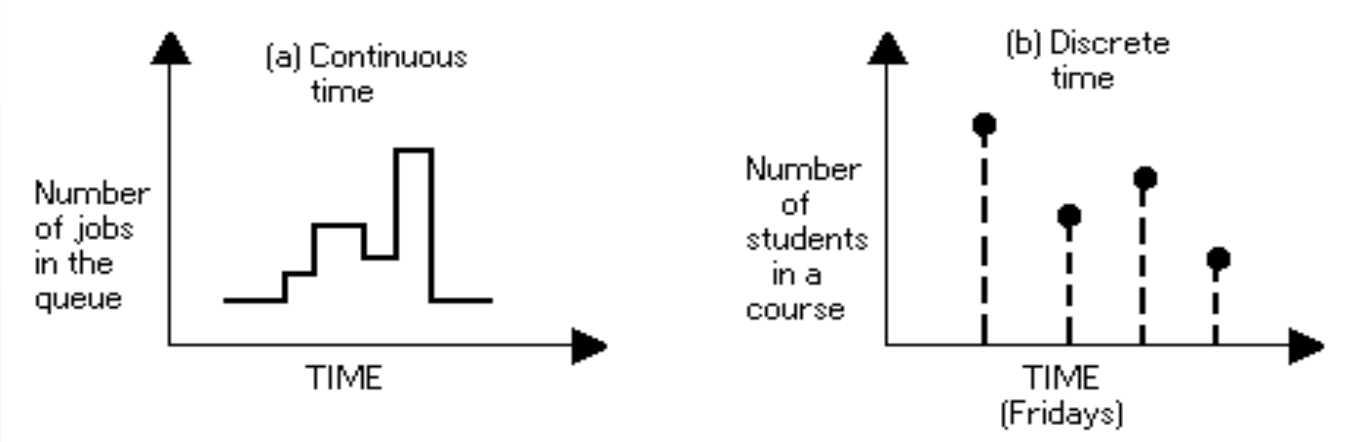
SIMULATION “WORLD-VIEWS”

- Pure Continuous Simulation
- Pure Discrete Simulation
  - Event-oriented
  - Activity-oriented
  - Process-oriented
- Combined Discrete / Continuous Simulation

Examples Of Both Type Models

- Continuous Time and Discrete Time Models:

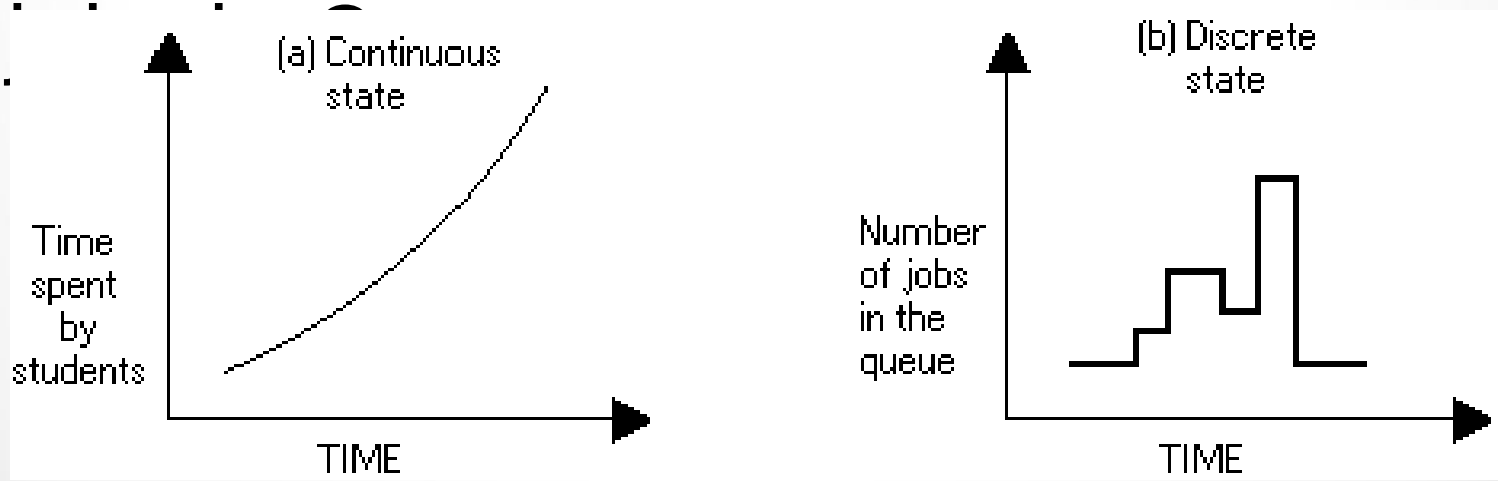
  CPU scheduling model vs. number of students attending the class.



Examples (continued)

- Continuous State and Discrete State Models:

  Example: Time spent by students in a weekly class vs. Number of



Stochastic vs. Deterministic

- In deterministic models, the output of the model is fully determined by the parameter values and the initial conditions.
- Stochastic models possess some inherent randomness. The same set of parameter values and initial conditions will lead to an ensemble of different outputs.
- Obviously, the natural world is buffeted by stochasticity. But, stochastic models are considerably more complicated.

MODEL THE APPROPRIATE LEVEL(S) OF DETAIL

- Define the boundaries of the system to be modeled.
- Some characteristics of “the environment” (outside the boundaries) may need to be included in the model.
- Not all subsystems will require the same level of detail.
- Control the tendency to model in great detail those elements of the system which are well understood, while skimming over other, less well - understood sections.

START EARLY TO COLLECT THE NECESSARY INPUT DATA

Data comes in two quantities:

TOO MUCH!!

TOO LITTLE!!

With too much data, we need techniques for reducing it to a form usable in our model.

With too little data, we need information which can be represented by statistical distributions.

PROVIDE ADEQUATE AND ON-GOING DOCUMENTATION

In general, programmers hate to document. (They love to program!)

Documentation is always their lowest priority item. (Usually scheduled for just after the budget runs out!)

They believe that “only wimps read manuals.”

What can we do?

- Use self-documenting languages
- Insist on built-in user instructions(help screens)
- Set (or insist on) standards for coding style

DEVELOP PLAN FOR ADEQUATE MODEL VERIFICATION

Did we get the “right answers?”

Simulation provides something that no other technique does:

Step by step tracing of the model execution.

This provides a very natural way of checking the internal consistency of the model.

DEVELOP A PLAN FOR MODEL VALIDATION

VALIDATION: “Doing the right thing”

Or “Asking the right questions”

How do we know our model represents the

system under investigation?

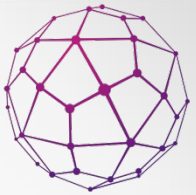
- Compare to existing system?
- Deterministic Case?

DEVELOP A PLAN FOR STATISTICAL OUTPUT ANALYSIS

- How much is enough?

  Long runs versus Replications

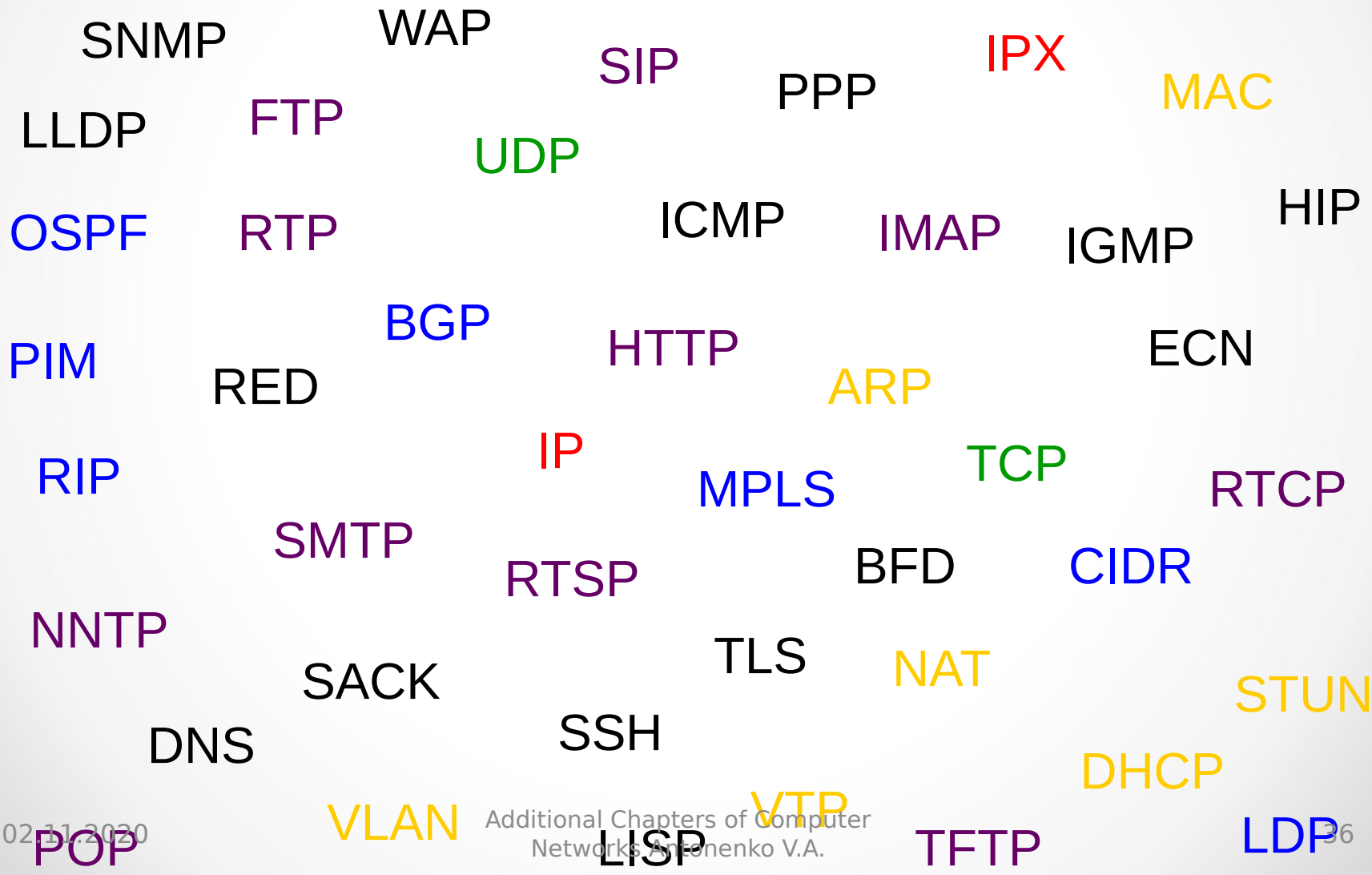
- Techniques for Analysis



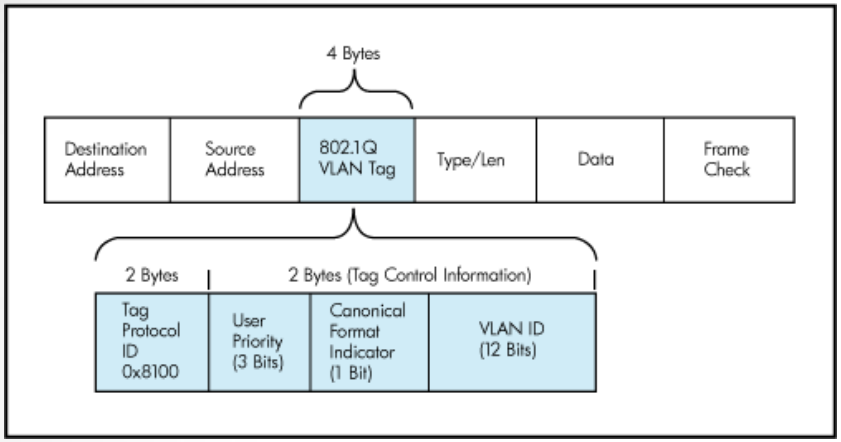
APPLIED RESEARCH CENTER FOR COMPUTER NETWORKS

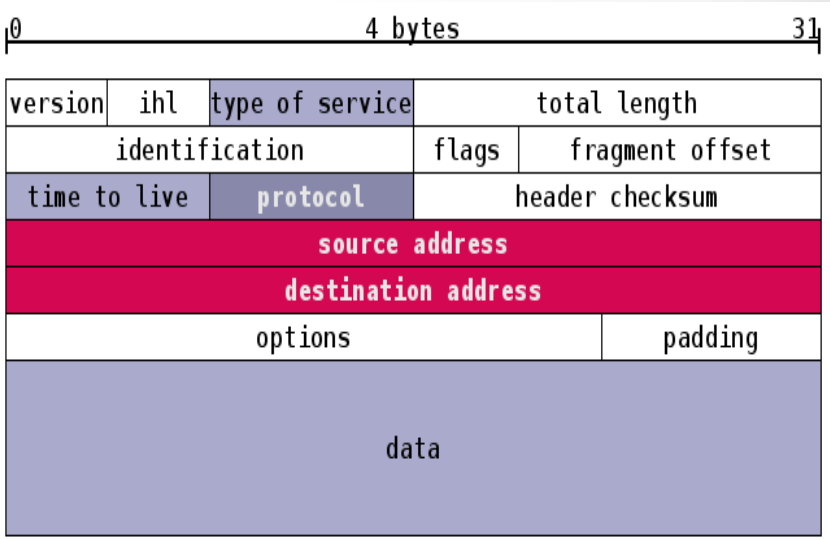
But what is networking?

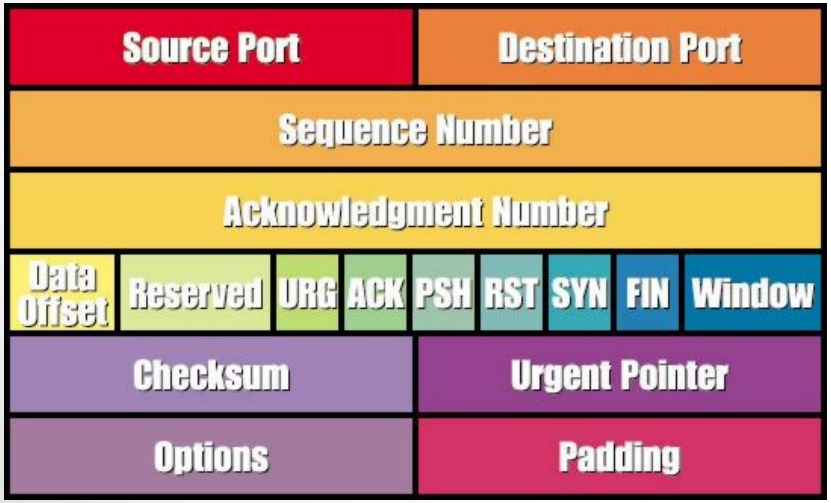
A Plethora of Protocol Acronyms?



A Heap of Header Formats?



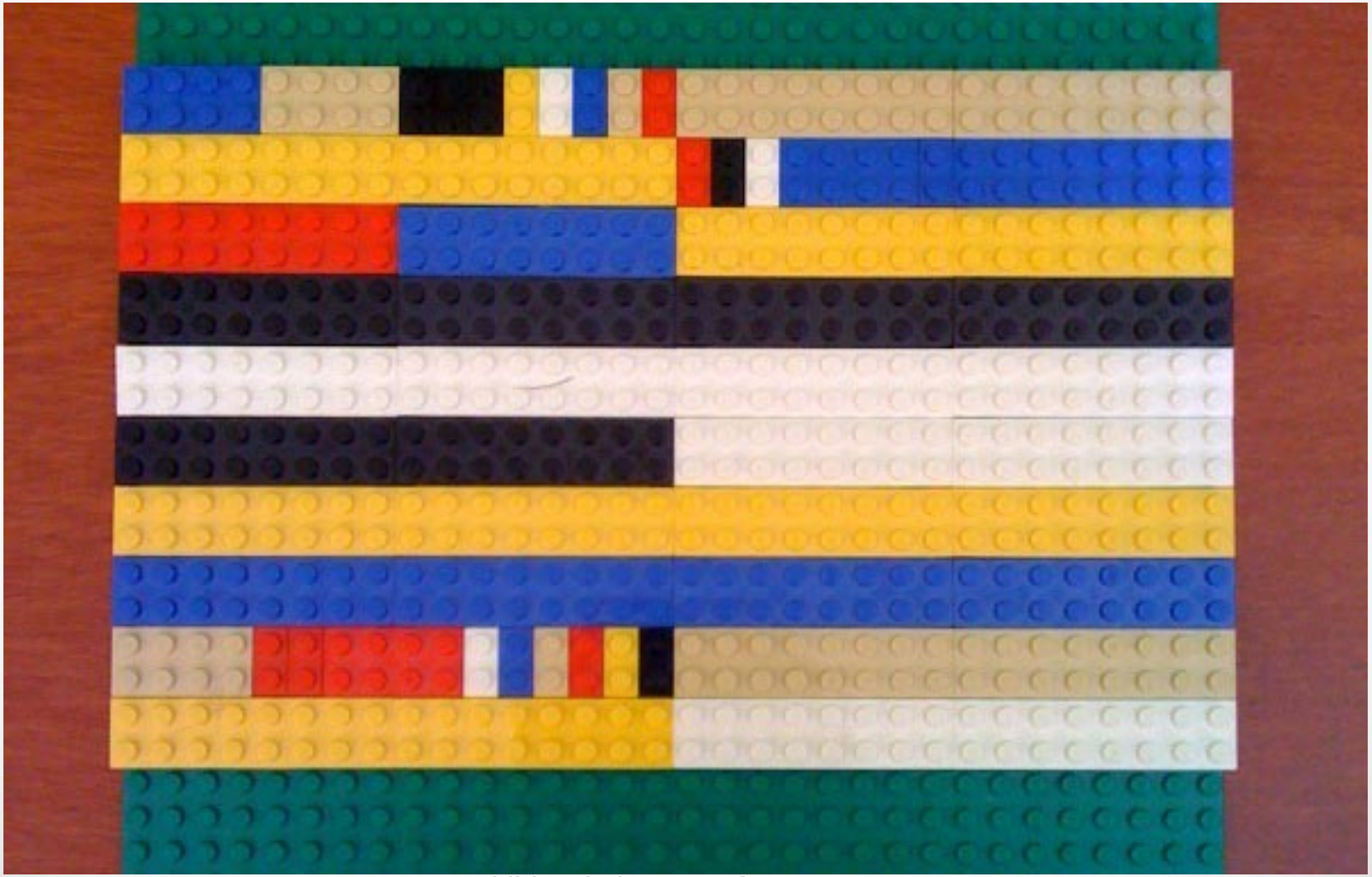




HTTP Response Header

Name	Value
HTTP Status Code: HTTP/1.1 200 OK	
Date:	Thu, 27 Mar 2008 13:37:17 GMT
Server:	Apache/2.0.55 (Ubuntu) PHP/5.1.2
Last-Modified:	Fri, 21 Mar 2008 13:57:30 GMT
ETag:	"358a4e4-56000-ddf5c680"
Accept-Ranges:	bytes
Content-Length:	352256
Connection:	close
Content-Type:	application/x-msdos-program

TCP/IP Header Formats in Lego



A Big Bunch of Boxes?

Router

Label Switched Router

Load balancer

Switch

Scrubber

Repeater

Gateway

Intrusion Detection System

Bridge

Route Reflector

Deep Packet Inspection

DHCP server

Packet shaper

Firewall

NAT

Hub

Packet sniffer

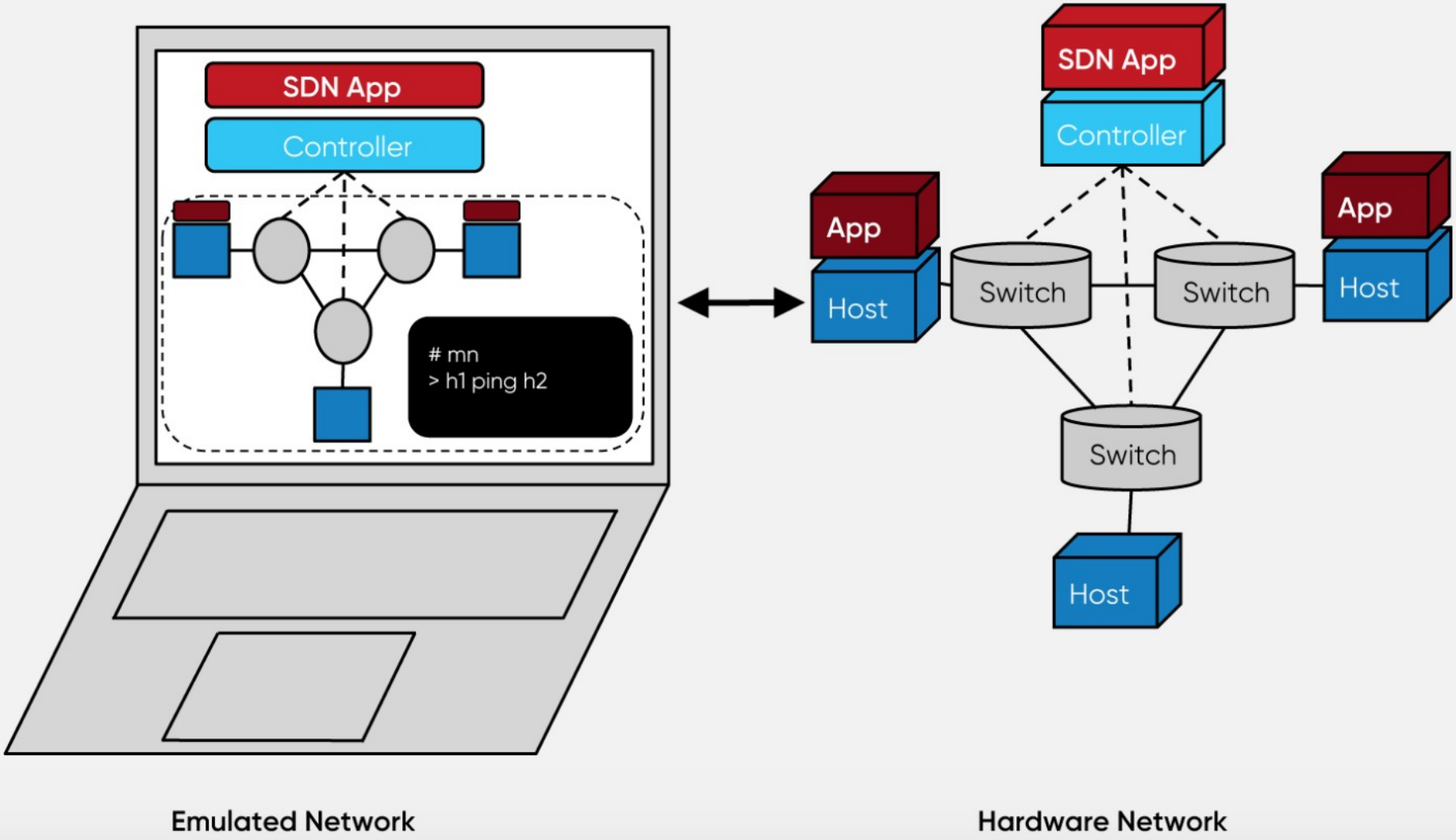
WAN accelerator

DNS server

Base station

Proxy

What Is Mininet?



What Is Mininet?

- ISOLATED HOSTS
  - A group of user-level processes moved into a network namespace that provide exclusive ownership of interfaces, ports and routing tables.

- EMULATED LINKS
  - Linux Traffic Control (tc) enforces the data rate of each link to shape traffic to a configured rate. Each emulated host has its own virtual Ethernet interface(s).

- EMULATED SWITCHES
  - The default Linux Bridge or the Open vSwitch running in kernel mode is used to switch

Mininet Installation

- Download the Mininet VM image.
- Download and install a virtualization system. We recommend VirtualBox (free, GPL) because it is **free** and works on OS X, Windows, and Linux (though it's slightly slower than VMware in our tests.) You can also use Qemu for any platform, VMware Workstation for Windows or Linux, VMware Fusion for Mac, or KVM (free, GPL) for Linux.
- Run through the VM Setup Notes to log in to the VM and customize it as desired.

…

- Or just do: sudo

APPLIED RESEARCH CENTER FOR COMPUTER NETWORKS

What Mininet Used For?

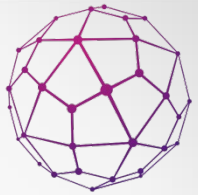
- Provides a simple and inexpensive **network testbed** for developing OpenFlow applications
- Enables **multiple concurrent developers** to work independently on the same topology
- Supports **system-level regression tests**, which are repeatable and easily packaged
- Enables **complex topology testing**, without the need to wire up a physical network
- Includes a **CLI** that is topology-aware and OpenFlow-aware, for debugging or running network-wide tests
- Supports **arbitrary custom topologies**, and includes a basic set of **parametrized topologies** is **usable out of the box** without programming, but also Provides a straightforward and extensible **Python API** for network creation and experimentation

Why It Is Better?

- Compared to full system virtualization based approaches, Mininet:
  - **Boots faster**: seconds instead of minutes
  - **Scales larger**: hundreds of hosts and switches vs. single digits
  - **Provides more bandwidth**: typically 2Gbps total bandwidth on modest hardware
  - **Installs easily**: a prepackaged VM is available that runs on VMware or VirtualBox for Mac/Win/Linux with OpenFlow v1.0 tools already installed.
- Compared to hardware testbeds, Mininet
  - is **inexpensive** and **always available** (even before conference deadlines)
  - is **quickly reconfigurable and restartable**
- Compared to simulators, Mininet
  - runs **real, unmodified code** including application code, OS kernel code, and control plane code (both OpenFlow controller code and Open vSwitch code)
  - easily **connects to real networks**
  - offers **interactive performance** - you can type at it

Limitation of Mininet

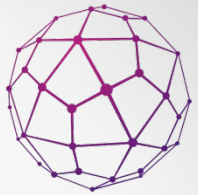
- Mininet-based networks cannot (currently) exceed the CPU or bandwidth available on a single server.
- Mininet cannot (currently) run non-Linux-compatible OpenFlow switches or applications; this has not been a major issue in practice.



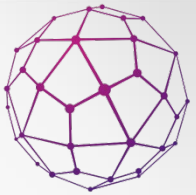
APPLIED RESEARCH CENTER FOR COMPUTER NETWORKS

- Start a minimal topology and enter the CLI:

- The default topology is the minimal topology, which includes one OpenFlow kernel switch connected to two hosts, plus the OpenFlow reference controller. This topology could also be specified on the command line with --topo=minimal. Other topologies are also available out of the box; see the --topo section in the output of mn -h.
- All four entities (2 host processes, 1 switch process, 1 basic controller) are now running in the VM. The controller can be outside the VM, and instructions for that are at the bottom.



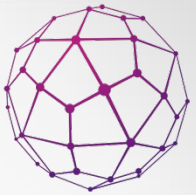
- Display Mininet CLI commands:
- Display nodes:
- Display links:
- Dump information about all nodes:



- If the first string typed into the Mininet CLI is a host, switch or controller name, the command is executed on that node.
- Run a command on a host process:

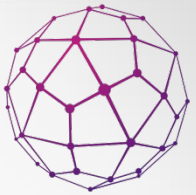
- You should see the host's h1-eth0 and loopback (lo) interfaces. Note that this interface (h1-eth0) is not seen by the primary Linux system when ifconfig is run, because it is specific to the network namespace of the host process.
- In contrast, the switch by default runs in the root network namespace, so running a command on the "switch" is the same as running it from a regular terminal:

- This will show the switch interfaces, plus the VM's connection out (eth0).



- Now, verify that you can ping from host 0 to host 1:

- If a string appears later in the command with a node name, that node name is replaced by its IP address; this happened for h2.
- An easier way to run this test is to use the Mininet CLI built-in pingall command, which does an all-pairs ping:



- Exit the CLI:

- If Mininet crashes for some reason, clean it up:

- Run a regression test:

- This command created a minimal topology, started up the OpenFlow reference controller, ran an all-pairs-ping test, and tore down both the topology and the controller.
- Another useful test is iperf (give it about 10 seconds to complete):

- This command created the same Mininet, ran an iperf server on one host, ran an iperf client on the second host, and parsed the bandwidth achieved.

- The default topology is a single switch connected to two hosts. You could change this to a different topo with --topo, and pass parameters for that topology’s creation. For example, to verify all-pairs ping connectivity with one switch and three hosts:

- Another example, with a linear topology (where each switch has one host, and all switches connect in a line):

- Parametrized topologies are one of Mininet’s most useful and powerful features.

- Mininet 2.0 allows you to set link parameters, and these can even be set automatially from the command line:

- If the delay for each link is 10 ms, the round trip time (RTT) should be about 40 ms, since the ICMP request traverses two links (one to the switch, one to the destination) and the ICMP reply traverses two links coming back.

- Custom topologies can be easily defined as well, using a simple Python API, and an example is provided in custom/topo-2sw-2host.py. This example connects two switches directly, with a single host off each switch.
- When a custom mininet file is provided, it can add new topologies, switch types, and tests to the command-line. For example:

$ sudo mn --cu … sw-2host.py --

```python
"""Custom topology example

Two directly connected switches plus a host for each switch:

   host --- switch --- switch --- host

Adding the 'topos' dict with a key/value pair to generate our newly defined
topology enables one to pass in '--topo=mytopo' from the command line.
"""

from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

- The --mac option is super-useful, and sets the host MAC and IP addrs to small, unique, easy-to-read IDs.
- Before:

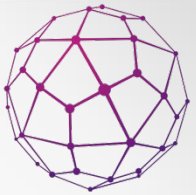
```
$ sudo mn
...
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr f6:9d:5a:7f:41:42
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:6 errors:0 dropped:0 overruns:0 frame:0
         TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:392 (392.0 B)  TX bytes:392 (392.0 B)
mininet> exit
```

- After:

```
$ sudo mn --mac
...
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr 00:00:00:00:00:01
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
mininet> exit
```

- Other switch types can be used. For example, to run the user-space switch:

- Another example switch type is Open vSwitch (OVS), which comes preinstalled on the Mininet VM. The iperf-reported TCP bandwidth should be similar to the OpenFlow kernel module, and possibly faster:



APPLIED RESEARCH CENTER FOR COMPUTER NETWORKS

- If the first phrase on the Mininiet command line is py, then that command is executed with Python. This might be useful for extending Mininet, as well as probing its inner workings. Each host, switch, and controller has an associated Node object.
- At the Mininet CLI, run:

- Print the accessible local variables:

- Next, see the methods and properties available for a node, using the dir() function:

- You can read the on-line documentation for methods available on a node by using the help() function:

- (Press "q" to quit reading the documentation.)
- You can also evaluate methods of variables:

- For fault tolerance testing, it can be helpful to bring links up and down.
- To disable both halves of a virtual ethernet pair:

- You should see an OpenFlow Port Status Change notification get generated. To bring the link back up:

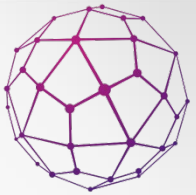
- When you start a Mininet network, each switch can be connected to a remote controller - which could be in the VM, outside the VM and on your local machine, or anywhere in the world.
- If you want to try this, fill in the host IP and/or listening port:

$ sudo n ntroller

- For example, to run POX's sample learning switch, you could do something like

- in one window, and in another window, start up Mininet to connect to the "remote" controller (which is actually running locally, but outside of Mininet's control):

$ sudo mn --co t,port=6633



APPLIED
RESEARCH
CENTER FOR
COMPUTER
NETWORKS

Thank You for Attention!