Protocol Design and The Real World

Protocol Design

Living below the Internet:

Advice for
Internet Subnetwork Designers

RFC 3819, July 2004
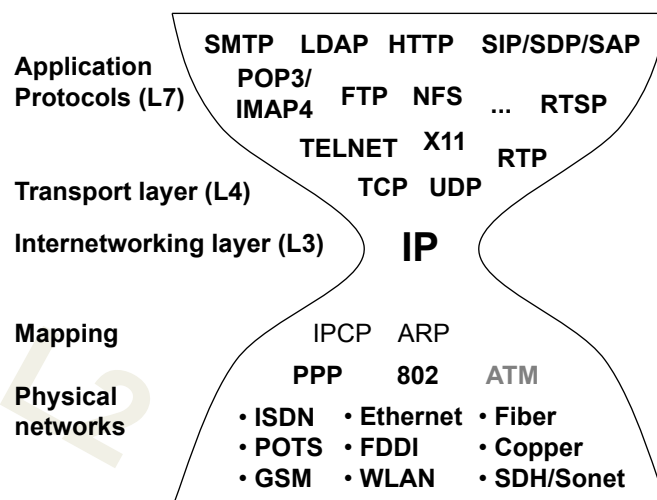
Internet: The Network of Networks

▶ Networking technologies come and go
  • Modem, ISDN, DSL, Cable modem, Sat modem, Fiber to the home, Hybrid Fiber Coax, Powerline, Wireless Local Loop/WiMax, WiFi, …

▶ Each of the technologies has some characteristic technical parameters:
  • Bitrate/data rate/throughput
  • Transmission latency (light speed!)
  • Range/coverage/availability
  • Cost!

▶ Moore's Law keeps shifting the design tradeoffs
  • More transistors allow more processing
  • And new technologies are invented every day

▶ IP must be able to interface to all of the network technologies
▶ But each of the subnetwork technologies can help or hurt with this

---

Internet Protocol "Suite"

Application Protocols (L7)

SMTP LDAP HTTP SIP/SDP/SAP
POP3/IMAP4 FTP NFS ... RTSP
TELNET X11 RTP

Transport layer (L4)

TCP UDP

Internetworking layer (L3)

IP

Mapping

IPCP ARP

PPP 802 ATM

Physical networks

• ISDN   • Ethernet   • Fiber
• POTS   • FDDI   • Copper
• GSM   • WLAN   • SDH/Sonet

What is a "subnetwork"?

▶ IP packets are carried by "links", "link layer", "L2"

▶ RFC 2460 defines "Link" as:
  - a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6.
    Examples are Ethernets (simple or bridged); PPP links; X.25, Frame Relay, or ATM networks; and internet (or higher) layer "tunnels", such as tunnels over IPv4 or IPv6 itself.

▶ A "Link" can be highly structured
  - Ethernets are connected by switches (= bridges) and formerly repeaters
  - Some "Links" are multi-layer networks, e.g. the serial line emulation defined by GSM runs its own mobility protocol

▶ IP generally does not care too much
  - But its performance can be helped or hurt

Optimizing subnetwork performance

▶ Provide functionality sufficient for carrying IP
  - Move IP packets back and forth
  - Provide some form of L3 → L2 address mapping

▶ Eliminate unnecessary functions that increase cost or complexity
  - IP does not need perfect retransmission persistence
  - Traditionally, subnetwork designers have erred on the side of **too much** functionality (remember the end-to-end arguments)
  - Waist-expanders (multicast, QoS) do benefit from L2 support

▶ Choose subnetwork parameters that maximize the performance of the Internet protocols
  - E.g., losses should be predominantly congestion losses

MTUs, fragmentation, segmentation (1)

▸ IPv4 has been designed to "work" with MTUs of 68 bytes

▸ Minimum **reassembly unit** was 576 bytes originally in IPv4

▸ Dominance of Ethernet has caused the expected MTU to be 1500 Bytes
- Often with some bytes taken away for tunneling, PPPoE etc.
- IPv6 formalizes this to a **minimum MTU** of 1280 bytes

▸ IP packets
- Carry their own length (unless header compression is used)
- Allow fragmentation at the router (IPv4) or at the sender (IPv6)
  - Typically avoided by "Path MTU discovery", so MTU should be stable
  - Internet fog may cause ICMP "packet too big" messages to be lost, though
- Have only 16 bits (IPv4) or 32 bits (IPv6) for fragment IDs

---

MTUs, fragmentation, segmentation (2)

▸ IPv6 links must, IPv4 links should attain 1280..1500 byte MTU
- May need adaptation layer for segmentation/reassembly
- Much more efficient to do on the link layer

▸ Larger MTUs (9000+) become increasingly desirable at high speeds
- Sometimes called "jumbograms" (these are really packets > 64KB)

▸ Slow network **may** benefit from smaller MTUs
- Serialization delay (1.25 s @ 9600 bit/s!) should not exceed 100..200 ms
- When large packets block high-priority ones:
  Suspend-resume schemes (e.g., RFC 2687) or brute-force segmentation with multiple reassembly queues (e.g., RFC 2686, ATM) can help

Framing

▶ L1 transports (groups of) bits, L2 builds frames

▶ Delimiters vs. counting
  - Delimiters: maintain data transparency by bit stuffing, byte stuffing, etc.
  - COBS (constant overhead byte stuffing) is good way of providing transparency

▶ Easiest case: 1:1 mapping of IP packets to L2 packets

▶ SAR (small fixed-size frames, as in ATM): avoid complexity
  - AAL5: SNDUs with IP packet, length, CRC are chopped up
  - Reassembly errors are caught in the CRC (and SNDU length)

▶ Where L2 already has (large) fixed-size frames: mix and match
  - RFC 4326 (ULE) defines one such mapping on MPEG-2 frames (188 bytes)
  - To avoid error propagation, resynchronization should be quick

---

L2 connection management

▶ L2 may need connections (e.g., POTS/ISDN!)

▶ Manual setup
  - Acoustic coupler, anyone? ☺

▶ Automatic setup:
  - Nailed-up (i.e., reconnect after each failure)
  - Dial-on-demand + idle timeout
    ▪ Timeout value hard to choose
  - Bandwidth-on-demand (multiple connections "as needed")
    ▪ "Need" hard to find out from L2 as there is no L7 intention signaling

▶ Related: connection-less BoD
  - DAMA (Demand-Assignment Multiple Access)
  - 802.11 PCF

Multipoint networks

▶ Simplest case: PPP — address resolution is trivial
   • As is multicast
▶ Broadcast networks
   • IPv4 ARP requires broadcast (designed for Ethernet)
   • May have efficient multicast (IPv6 ND relies on this)
   • Infrastructure (e.g., Ethernet switches) may have to do the work
   • IGMP/MLD snooping (or explicit signalling protocol) to minimize exposure to unwanted multicast
▶ NBMA (non-broadcast multiple access)
   • Need additional support for discovery/address resolution
   • E.g., ATM had ATMARP, MARS

11

Error Control

▶ Ultimate responsibility: hosts (end-to-end argument)
   • Internet has license to drop, corrupt, duplicate, or reorder packets
▶ End-to-end repair is more expensive, though:
   • requires effort at multiple hops
   • Can only happen at path RTT timescales (as opposed to hop RTT)
   • Losses are interpreted as congestion by L4 and reduce throughput
▶ L2 may repair errors **to aid performance**
▶ Actually: some loss is OK (or even needed!)
   • Perfect persistence will be overtaken by TCP retransmission
▶ L2 reliability should be "lightweight"
   • it only has to be "good enough"

12

Assessing L2 error control

▶ Yardstick: TCP
- Most traffic is TCP anyway
- Other traffic is supposed to be TCP-friendly (and generally have similar performance characteristics)

▶ Secondary consideration: RTP
- Looks different
- Has different requirements
  - **consistently** low delay keeps the playout timer short
  - Every packet drop reduces quality (but a couple percent can be tolerated)

▶ Two approaches to add redundancy:
- Always: Forward error correction (FEC), often at L1
- On demand: retransmission ("ARQ"), at L2

FEC

▶ From a total throughput perspective, worse than ARQ
- But for ARQ you first have to get entire packets (frames) through

▶ Now universally used at L1 (Trellis coding etc.)

▶ Issue: FEC vs. fading
- FEC requires interleaving to ride through deep fades
- Interleaving adds **delay**
- TCP performance inversely proportional to delay

▶ Modern thinking ("4G") : minimize delay
- Hop-by-hop ARQ works quite well on a low-delay channel
- Need to leave some spare capacity for retransmissions, though

ARQ

▶ RFC3366

▶ Hop-by-hop retransmission wins:
- Can operate on link-layer friendly segments (e.g., < 100 Byte)
- Involves only the resources of one hop
- Operates at the time constants of one hop

▶ Wild delay variation introduced by ARQ loses:
- TCP timers will fire ahead of time if ARQ takes too long
- Leads to duplicate packets — possibly both in the same L2 queue…

▶ Limit retransmission persistency
- Should be on the order of **path** delay
- Somewhat hard to predict (LAN vs. country vs. continent vs. world)
- If possible, distinguish TCP (higher persistency) and RTP (lower persistency)

Outages

▶ "Elevator events": system enters tunnel/metal cage/…

▶ TCP timers **will** fire
- No sense transmitting all the duplicate packets from multiple retransmissions
- High persistence not very useful
- **Do not deliver all the stale packets** after the outage

▶ However: There is no way in IP to notify the end of the outage
- TCP timers may have backed off into some high region
- It may take a while until the next timer fires
- Dead time after the end of the outage

▶ Trick: Keep **some** packets around at L2 during an outage
- Delivery after outage will trigger L4 machinery

Quality of error control

▶ TCP, UDP (as well as ICMP and IPv4 itself) use 16-bit two's complement checksum
  - Easy to compute (also by combining from partial checksums)
  - Not very strong — relies on good error detection at L2
  - Lots of undetected errors in practice [Stone/Partridge2000]
  - SCTP is the odd one out (CRC-32c, RFC3309)
▶ Higher layers can (and often do) use better error detection
  - E.g., cryptographic checksums in AH, ESP, TLS, SSH
▶ Still, some minimum quality from L2 is **expected**
  - Most L2 have at least 16-bit CRC
  - Make sure frame size and CRC are compatible
    ▪ Long frames should use 32-bit CRC
  - Doing this at packet level is better than at segment level (cf. AAL5)

Unequal error protection

▶ Some applications can tolerate errors in some of their data
  - E.g., GSM speech codec can tolerate bit errors in excitation signal
▶ Need to protect header information, though
▶ Idea: error-protect initial part, but not all of the packet
▶ UDP-Lite (RFC3828): partial payload protection
  - Indicate which part of the UDP payload contributes to checksum
    ▪ Reuses redundant UDP length field
▶ This separation is not visible at subnetwork layer
  - L2 error protection would need to make the same distinction
  - Could be divined by peeking at L4 header
▶ No L2 implementation yet

QoS

▸ Integrated Services have L2 mappings
(ISSLL — integrated services over specific link layers)
- Controlled-Load may be easy to attain; Guaranteed is much harder
- With a shared L2, also need to address admission control (reservation)
- Tspec may be quite useful for planning resource usage (intention signal)

▸ Differentiated Services
- PHB (per-hop behaviors) such as AF and EF again need to be mapped down to L2
- AF has multiple priorities (as well as the backwards-compatible class selectors)

▸ Related issue: Buffering and Active Queue Management (AQM)
- Provide adequate buffers
- Start dropping some packets before latency gets really big (RED)
  - Hard to configure, though

Asymmetric Links

▸ RFC3449

▸ Some links have higher bitrates in one direction than in the other
- ADSL
- Satellites: downlink vs. return channels
- Hybrid links built out of different technologies (Satellite + ISDN)

▸ Problem: When the ACKs don't fit into the return channel, forward channel is impaired
- 1500/40 = 37.5 (usually less due to additional overheads)

▸ ACK compression etc. can help

▸ PEP (performance enhancing proxy) may be required
- Can also assist TCP with other problems (high delay, high corruption error rate)

Summary: TCP in Extreme Networks
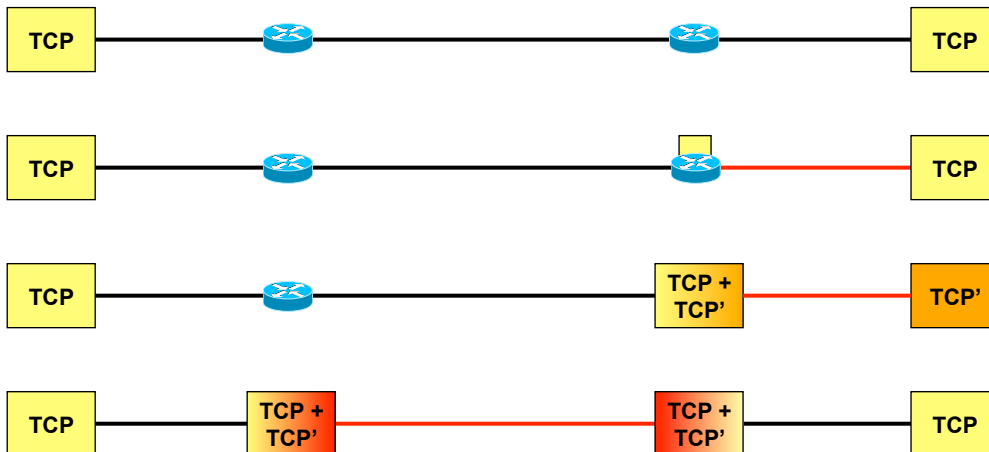
Compression

▸ Applications can compress their data
- SSH
- HTTP Content-Encoding
- GIF, JPEG, PNG, video formats…
- Very useful **before** encryption

▸ Many don't ➡ potential for performance increase at L2
- Hard to do efficiently without sequencing/retransmission, though
- Don't expand if L4 already compressed and/or encrypted

▸ Similar: Header compression
- Most beneficial at small MTUs or for small-packet data (RTP voice)
- Hop-by-hop can compress IP (and L4) headers, too
- Needs to cope with packet losses, possibly reordering

Reordering

▶ IP allows for reordering of packets

▶ TCP, however, loses performance if that happens
- May mis-diagnose a packet loss (three dup-acks)

▶ RTP, properly implemented, can be quite happy with reordering
- As long as the timescales do not diverge too much

▶ Try to avoid reordering
- As long as it does not impair performance

▶ Many L2 protocols also expect in-order delivery
- PPP only works on order-preserving links
- Existing header compression schemes: see RFC4224

L2 Security

▶ L2 security can
- Protect the network (where its operation is expensive)
  - And protect against theft of service via that specific L2 network
- Equalize security to other parts of the network
  - I.e., protection against casual snooping may be all a user wants
- Thwart traffic analysis

▶ L2 security cannot really:
- Protect the radio resources (jammers are easy to build)
- Provide end-to-end security

From Specification to the Real World

Protocol Design

You have designed a protocol – what now?

▶ Implement it
- Good idea
  - Shows that you can implement it
  - And gives a clear idea how complex it really is
  - You will find errors, omissions, and ambiguities only when implementing
  - "Rough consensus and running code"
- But requires a lot of effort
  - You may want to do partial validation with less effort early on
- Errors in the spec: you may have to write parts over and over again
- An implementation by itself does not tell you much
  - About the scalability of your protocol: what happens if many nodes run it?
  - About its reliability, robustness, and performance in the Internet
- [An implementation alone is often insufficient for publications]
  - You need to "prove" your ideas right
  - You need to deliver some quantitative data ("plots") that show you are better in some way

Alternatives?

▶ Analysis: mathematical modeling and quantitative evaluation
- Depends on your math skills and experience
- Of course, you should always do the minimal math yourself
  - Basic thoughts on scalability, etc.
- Anything coming close to the real world likely to get really complex
- Not in our focus

▶ Simulation: test your algorithms in an artificial environment
- Takes the place of real-world validation
- Requires some "implementation" in a simulator
  - Most ideas never make it beyond this step
- Often this is as close as you can get to real world experience

▶ Emulation: run your implementation in an artificial environment

---

Simulations

▶ There are many tools out there
  General purpose examples:
- ns-2/ns-3   [http://www.isi.edu/nsnam/ns/]
- GloMoSim   [http://pcl.cs.ucla.edu/projects/glomosim/]
- OMNET++   [http://www.omnetpp.org/]
- OPNET   [http://www.opnet.com]
- QualNet   [http://www.scalable-networks.com/]
- CSIM   [http://www.atl.external.lmco.com/projects/csim/]
- MIRAI-SF   [http://mirai-sf.nict.go.jp/index_e.html]
- MATLAB/Mathematica
- (Spreadsheets…)
  Special purpose tools for specific simulation environments
  (and there are many community efforts and extensions available)
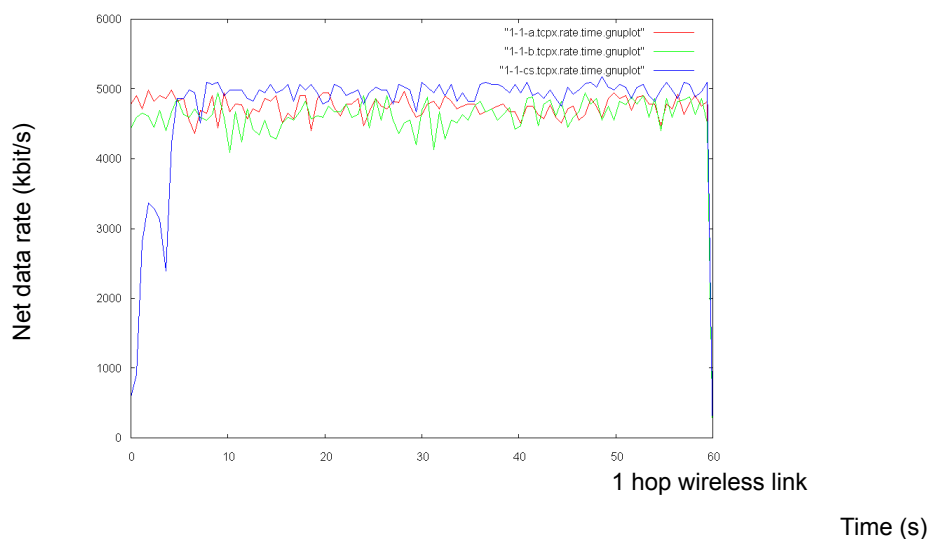
Issue #1 with Simulations

▶ Relation to Reality!

▶ Link layer: example wireless communication
- Radio propagation has a gazillion dependencies
- You cannot capture all
- You cannot model all potential sources of interference
  - People, opening and closing doors, carried laptops and mobile phones (Bluetooth), etc.
  - Furniture, wall and window characteristics, water on windows, etc.
  - Vehicles (trucks with different loads and shapes vs. full buses vs. empty buses vs. different cars vs. motorcycles vs. bicycles) at different velocities, densities
  - Density of buildings, types of buildings, park areas, …
  - Non communication interference: micro wave ovens, …

▶ Mobile communications: reasonable mobility models
- "Random waypoint considered harmful" — and indeed it is
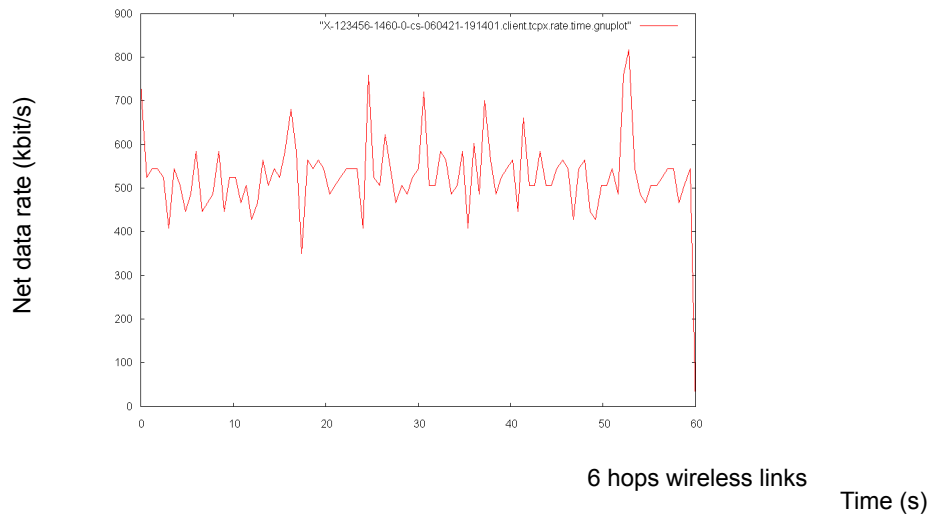- General issue: how do humans, vehicles, etc. move?
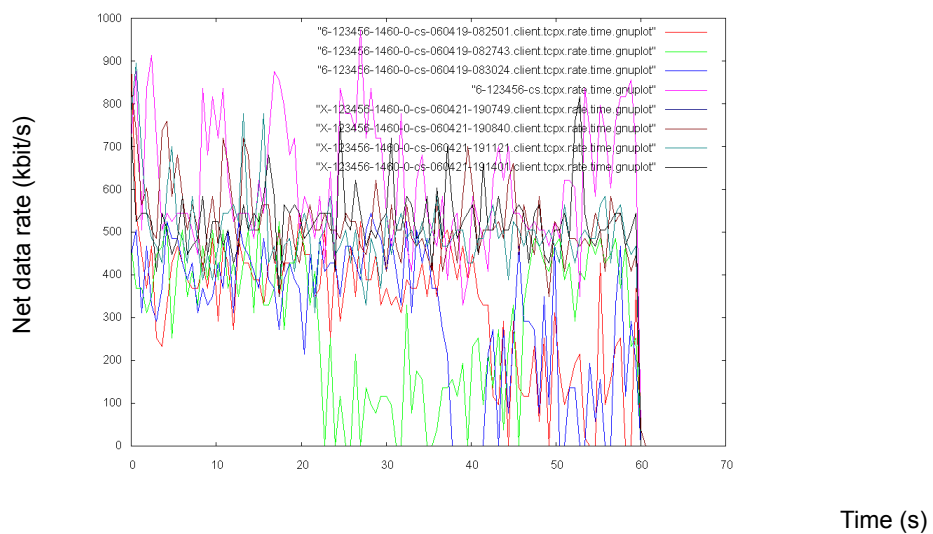

Variation in Wireless Links (1)

1 hop wireless link

Time (s)

Variation in Wireless Links (2)

6 hops wireless links

Time (s)

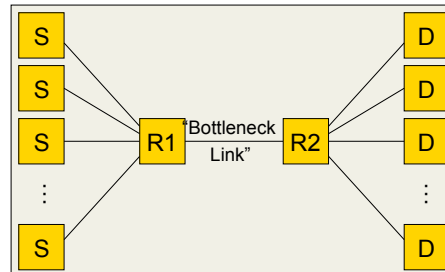# Variation in Wireless Links (3)

Time (s)

6 hops wireless links

Issue #1 with Simulations (2)

Network layer

▶ Internet complexity
- Interconnection topology
  - Networks, links, hosts
- Virtually impossible to model even parts

▶ Internet diversity
- Link data rates
- Routers
  - Queue sizes, queuing disciplines
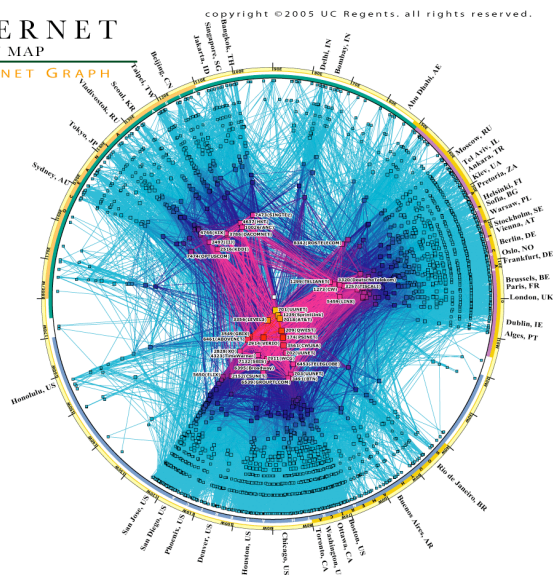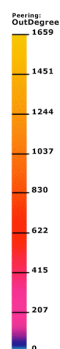 - General behavior
- Routing protocols

▶ "Background traffic"

Issue #1

▶ Network layer
- Internet:

Issue #1 with Simulations (3)

▶ Transport layer
- What mix of TCPs will you really find
- How much SCTP?
- How much UDP and similar traffic

▶ Applications
- Which application are run?
- Where?
  - Might be able to define this for web servers. But what about the others?
- What is the usage pattern?
  - Ratio between applications?
  - Behavior of an individual user or a group of users?
  - Variation over time?
  - New applications?
  - What is the resulting traffic?
- How well can large numbers help here?

Further Pitfalls with Simulations

▶ Simulating itself is tricky
- Find the right simulator, topology, traffic model, …
  - Difficult at all layers (virtually impossible for L1!)
- Need to implement your protocol in the simulator
  - Different constraints from the real-world
  - Does it match your real-world implementation?
- Choose the right simulation parameters

▶ Document everything
- Recommended reading: "MANET Simulation Studies: The Incredibles"

▶ After all: simulations are like statistics
- Don't trust any statistics you did not fake yourself!
- For others' results: be critical
- For your own simulations, this is like testing
  - Choose environments that are "real" and meaningful
    (rather than a perfect fit for what you want to prove)
  - Also choose environments that are "hostile"

Emulations

▶ Run the real code in a virtual environment
- Allows testing the real thing
  - Instead of some imitation for a simulator

▶ Few simple examples
- Dummynet   [http://info.iet.unipi.it/~luigi/ip_dummynet/]
- NIST Net   [http://snad.ncsl.nist.gov/nistnet/]
- Linux TCNG   [http://tcng.sourceforge.net/] [http://lartc.org/]
- Link layer packet bridges
- Simple traffic shaping tools (such as udppipe)

▶ Virtual network environments
- Virtualization of hosts (including kernel, interfaces, applications, etc.)
- May use real and/or virtual links
- May create complex artificial setups (similar to simulators)
- But run real code

▶ Obviously, some issues similar to simulations apply