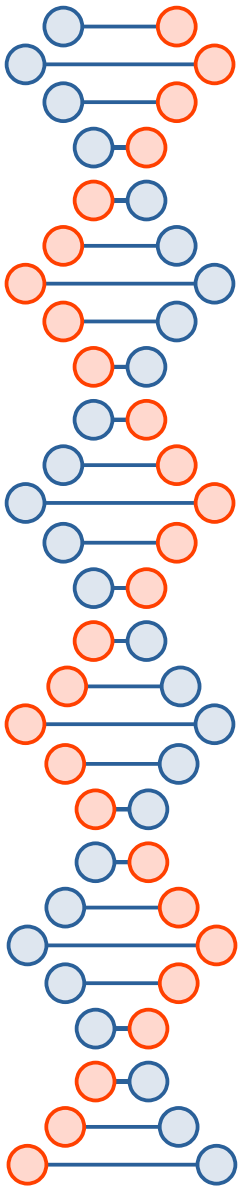
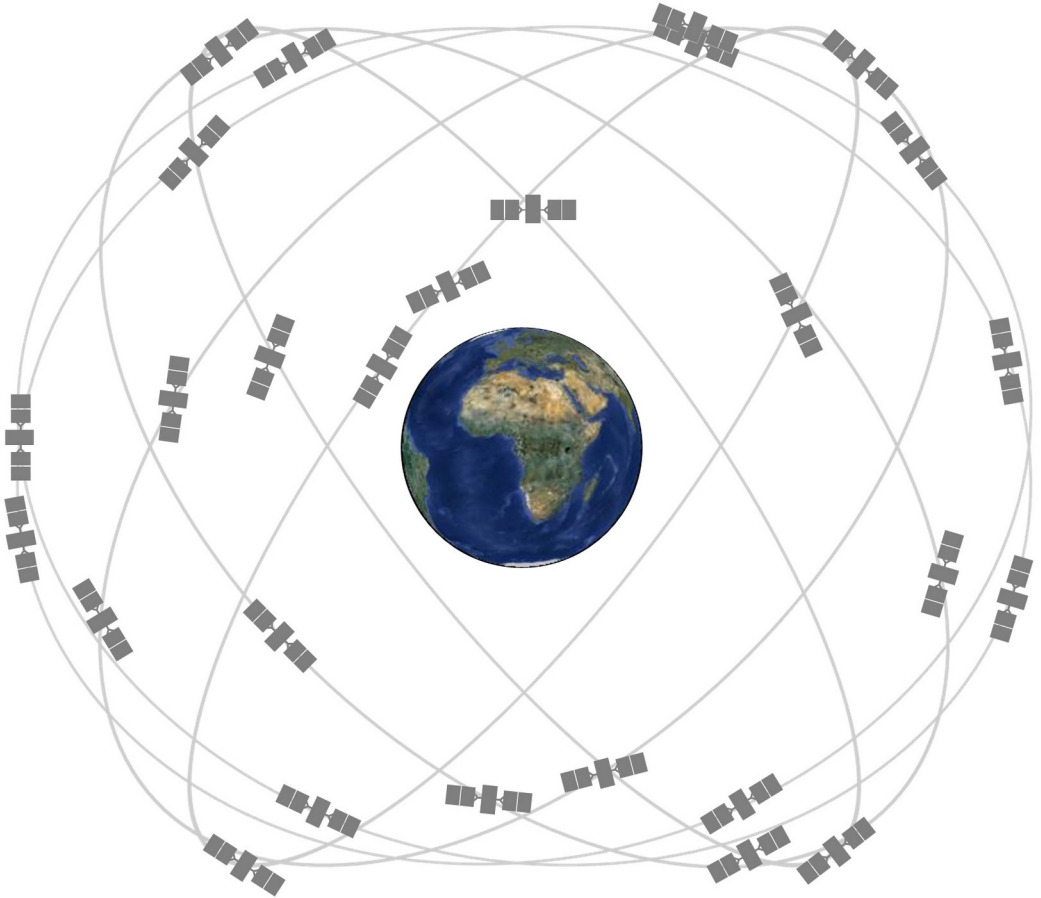


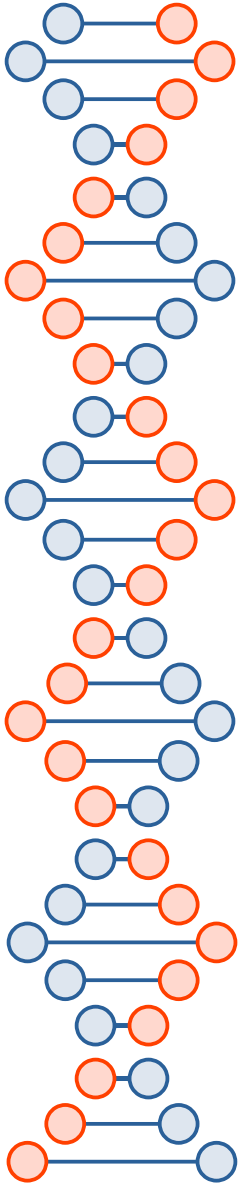
Intro to NIDS and NIDS evasion

CSE 468 Fall 2023

jedimaestro@asu.edu

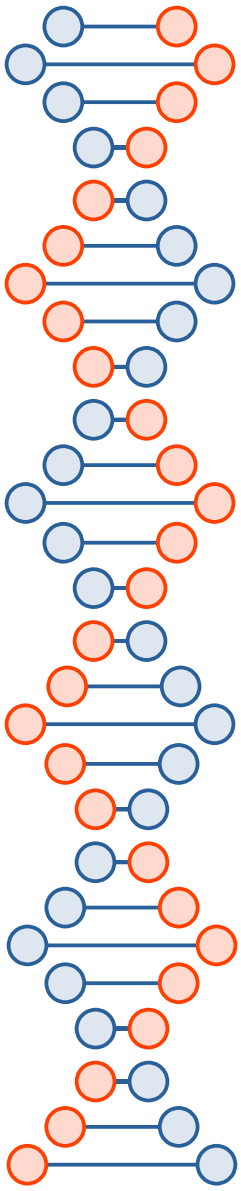
GPS (Global Positioning System)





GPS facts

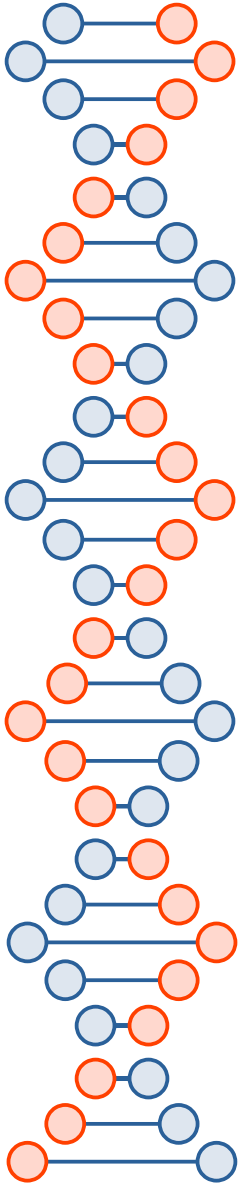
- Altitude of satellites is *approx.* 12,550 miles
- Moving about 7,000 miles per hour
  - At the equator, earth spins at about 1,000 miles per hour
- GPS signals reach earth in about 1/15th of a second
  - Going about 670,616,629 miles per hour
    - Every 1000 mph is about 0.000149116% error
      - Who cares?



GPS facts

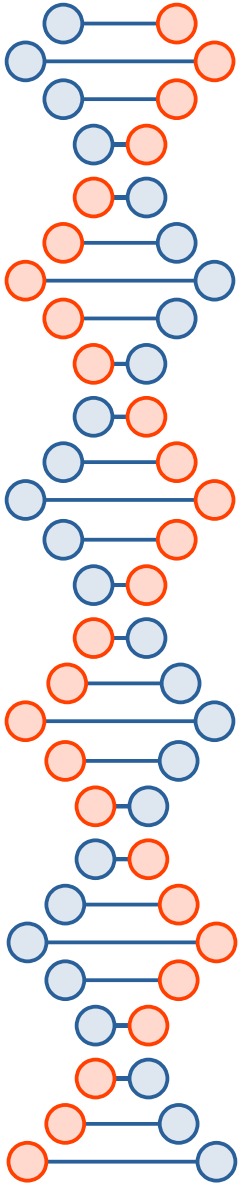
- Altitude of satellites is *approx.* 12,550 miles
- Moving about 7,000 miles per hour
  - At the equator, earth spins at about 1,000 miles per hour
- GPS signals reach earth in about 1/15th of a second
  - Going about 670,616,629 miles per hour
    - Every 1000 mph is about 0.000149116% error
      - Who cares?

0.000149116% of 12,550 miles is about 100 feet!



GPS corrections for velocities of satellites and Earth's spin?

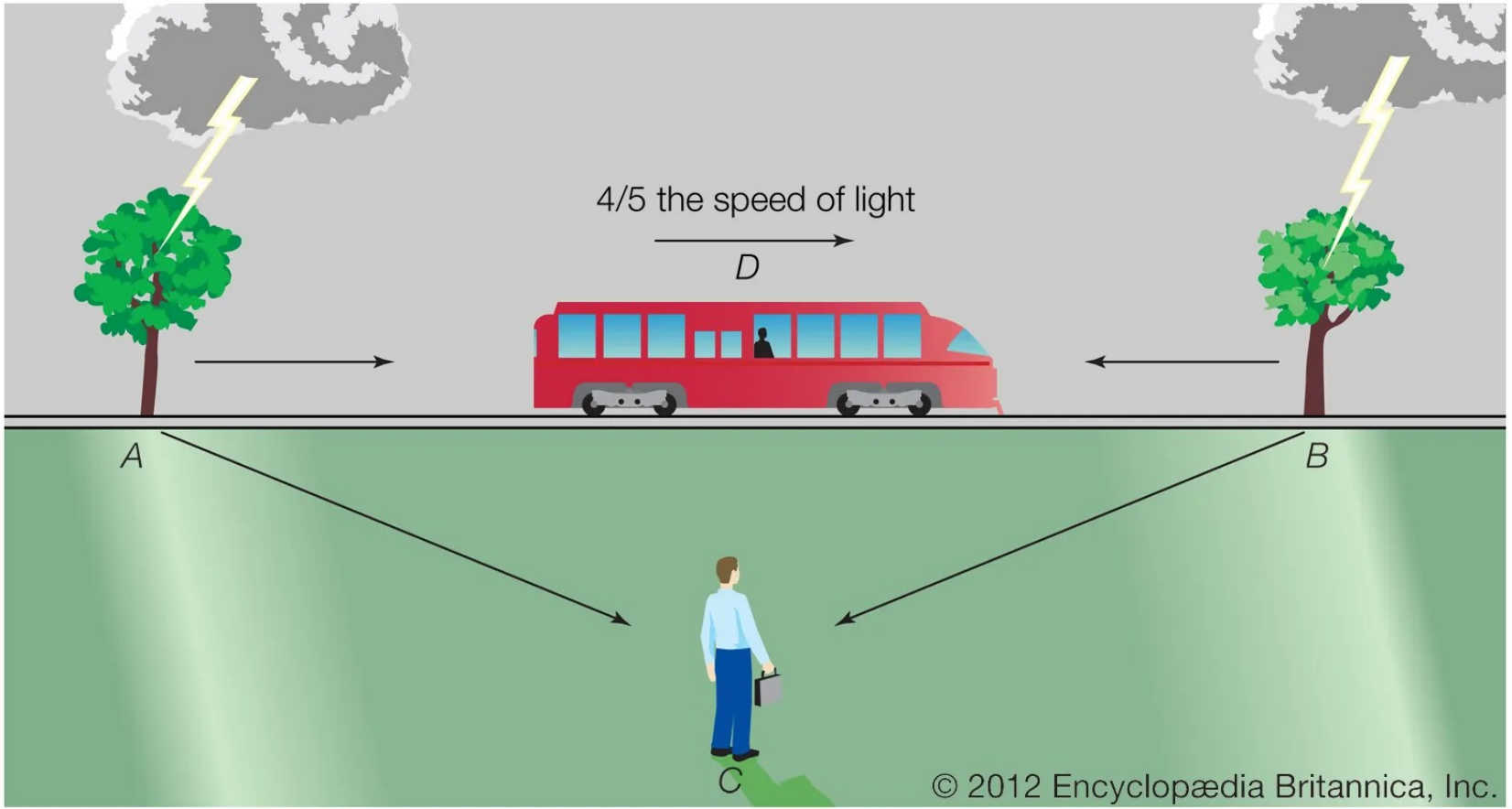
- If we're spinning 1000mph in one direction and the GPS satellite is going 7000mph in the other direction, *vs.*…
- we're spinning 1000mph and the satellite is going in the same direction at 7000mph (or orthogonal? an angle?)
- 1000 feet is a lot of error, compared to Coor classrooms 1000 feet away is like Biodesign B
- Do you think the software in your phone accounts for these errors?

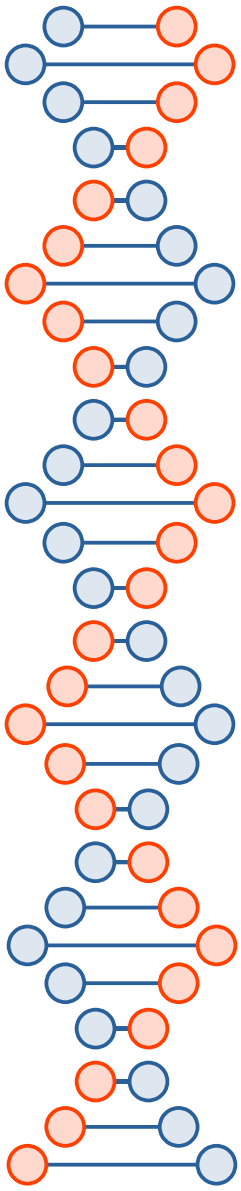


GPS corrections

- No correction for the velocities of satellites and the spin of the Earth
  - Einstein's theory of relativity
- Two corrections to the timers aboard the satellites combine for (slowed by a net of 38 microseconds per day)
  - Special relativity → satellites moving faster → time dillation → time is 7 microseconds per day slower
  - General relativity → satellites farther from Earth's gravity → time is 45 microseconds per day faster

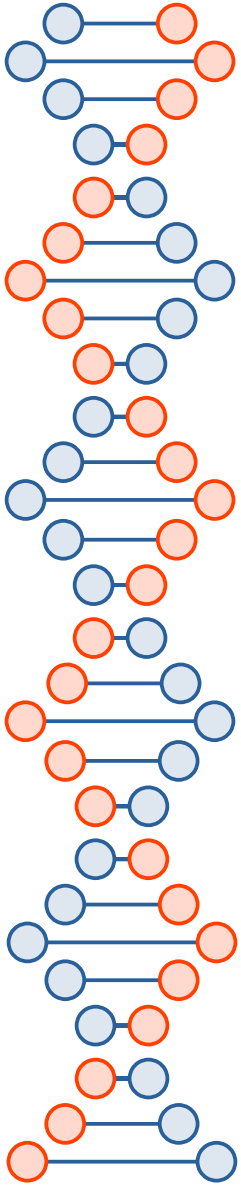
https://www.britannica.com/science/relativity/Special-relativity



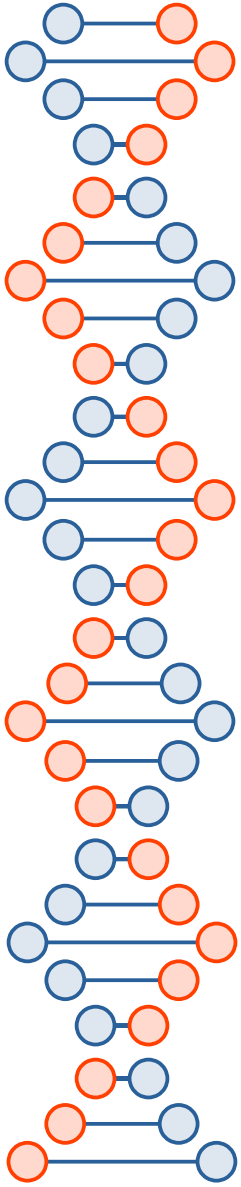


The universe is weird...

- Even the best clocks (*e.g.*, atomic clocks) get out of synch
  - *E.g.*, because of elevation
- Events are only only partially ordered
  - Events are relative to an observer
- Our networks and NIDS systems exist in this weird universe
  - Even if we ignored packet loss and variable network delay, assumed every computer/device had an atomic clock, and accounted for the rough elevation of devices there is no way for a NIDS to know for certain if, *e.g.*, a message was received before a connection timed out on one end of a network flow.



In other words, the physics is against us before we even start talking about computer networking concepts. When we start talking about the Internet, things get even worse. (This is good news for anti-censorship efforts, bad news for network admins who rely on NIDS).



NIDS

- Network Intrusion Detection System
- Dual use technology
  - Security (*e.g.*, filtering for malware)
  - Information controls (*e.g.*, censorship)
- Deep Packet Inspection (DPI)
- Are there more middleboxes on the Internet than there are routers?

UNIX process hierarchy

pstree

pstree -u jedi

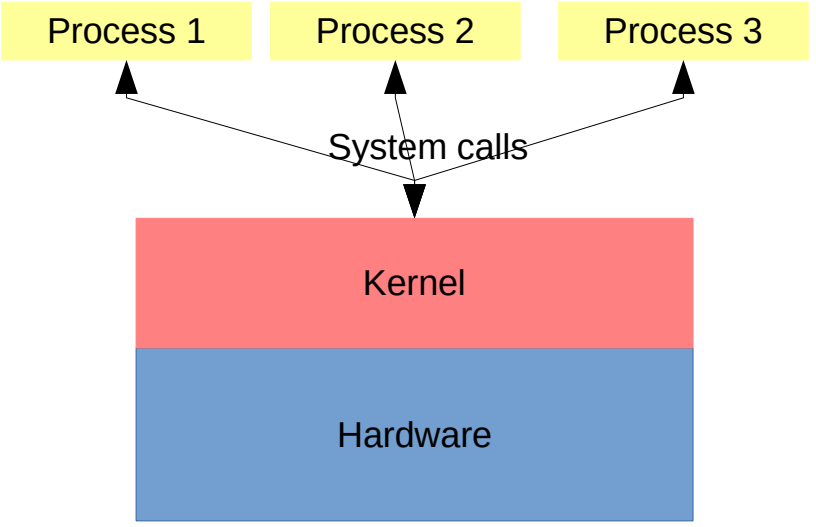
cd /tmp

wget phrack.org

less index.html

strace -f -o bla.txt wget phrack.org

less bla.txt



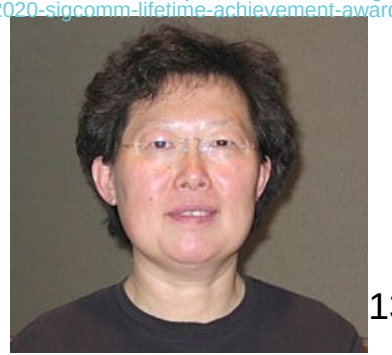
OSI model

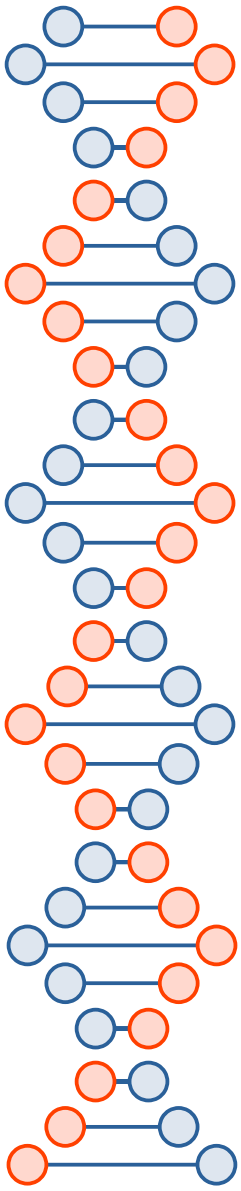
- Layer 1: Physical (think Ethernet, 802.11)
- Layer 2: Data Link (think ARP)
- Layer 3: Network (think IP)
- Layer 4: Transport (think TCP)
- Layer 5: Session (think NetBIOS, SOCKS)
- Layer 6: Presentation (think SSL/TLS)
- Layer 7: Application (think HTTP)

Lixia Zhang (张丽霞)

- Coined the term “middlebox” in 1999
- Jonathan B. Postel Professor of Computer Science at the University of California, Los Angeles
- One of 21 participants at the first IETF (Internet Engineering Task Force) meeting, only woman and only student
- Also… resource reservation, named data networking, driving tractors...
- https://en.wikipedia.org/wiki/Lixia_Zhang

https://www.cs.ucla.edu/professor-lixia-zhang-wins-the-2020-sigcomm-lifetime-achievement-award/

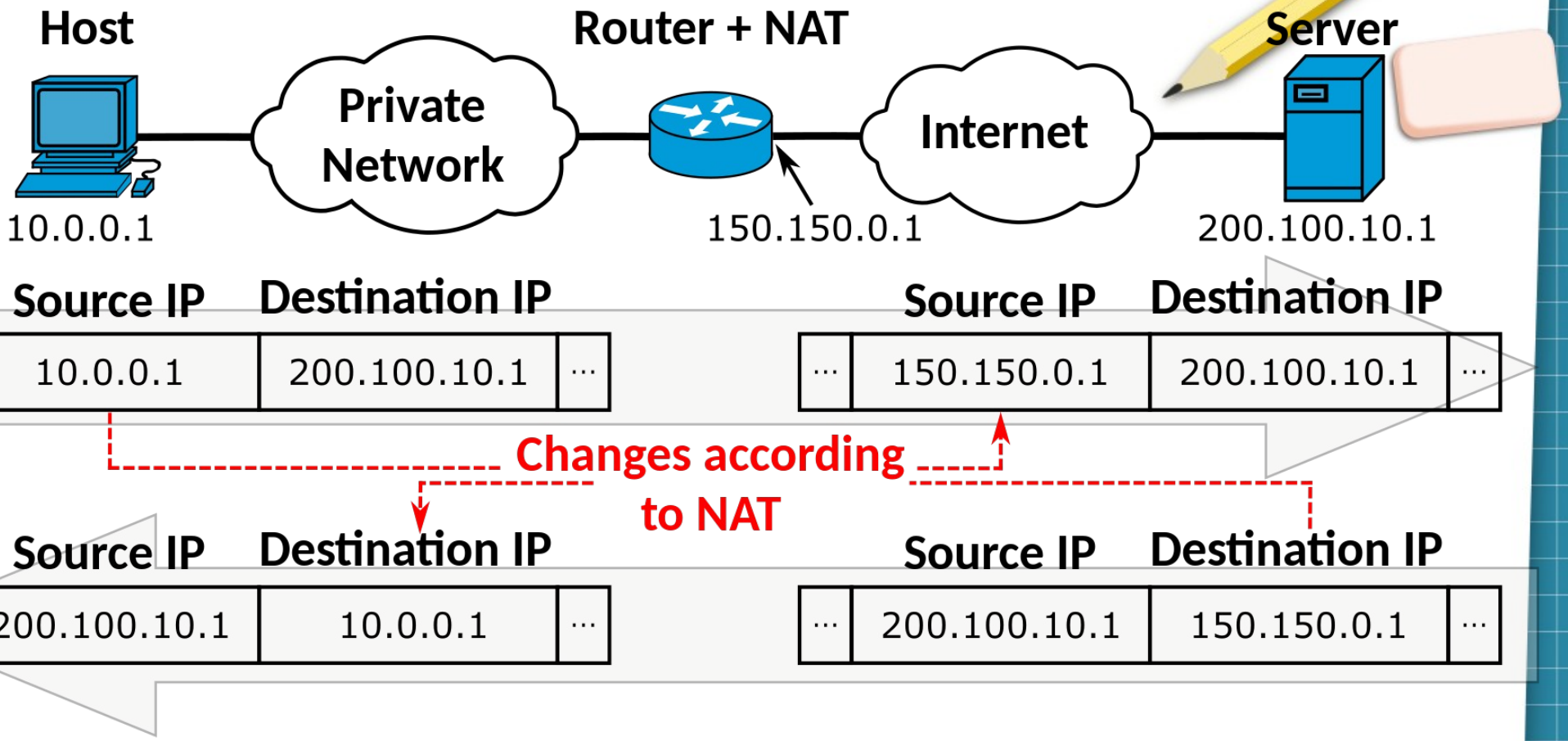




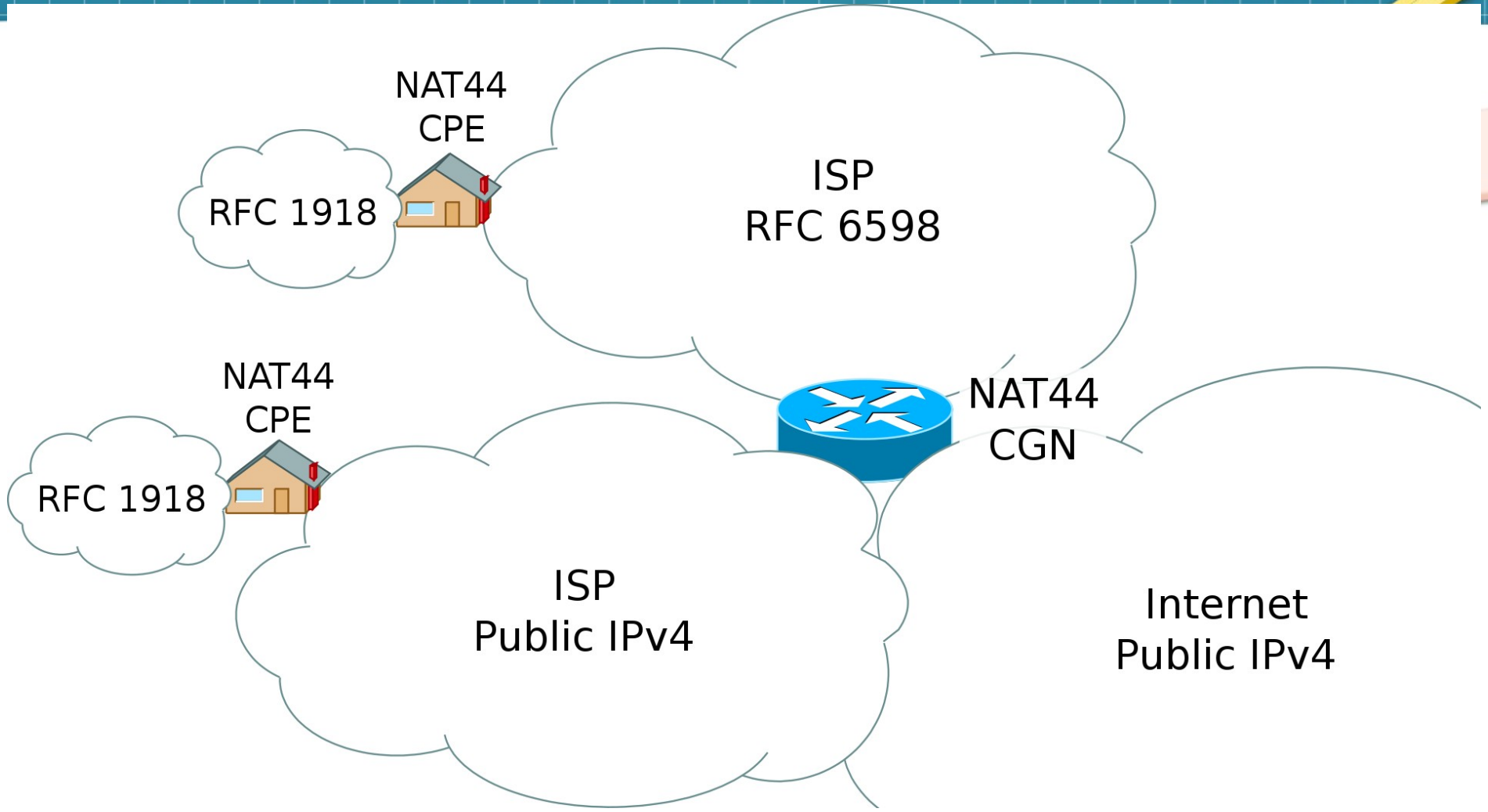
Examples of middleboxes...

NAT == Network Address Translation

- Typically between private and public
  - 192.168.0.0/16, 10.0.0.0/8, and 172.16.0.0/12 are private
  - 127.0.0.0/24 is loopback
  - Most of everything else is public (*i.e.*, routable)
- Bogon filtering
  - Internet routers drop packets to/from private IPs (mostly)
- Most of the times you use the Internet you're going through multiple layers of NAT
  - *E.g.*, access points, VPNs, Carrier Grade NAT (CGNAT)

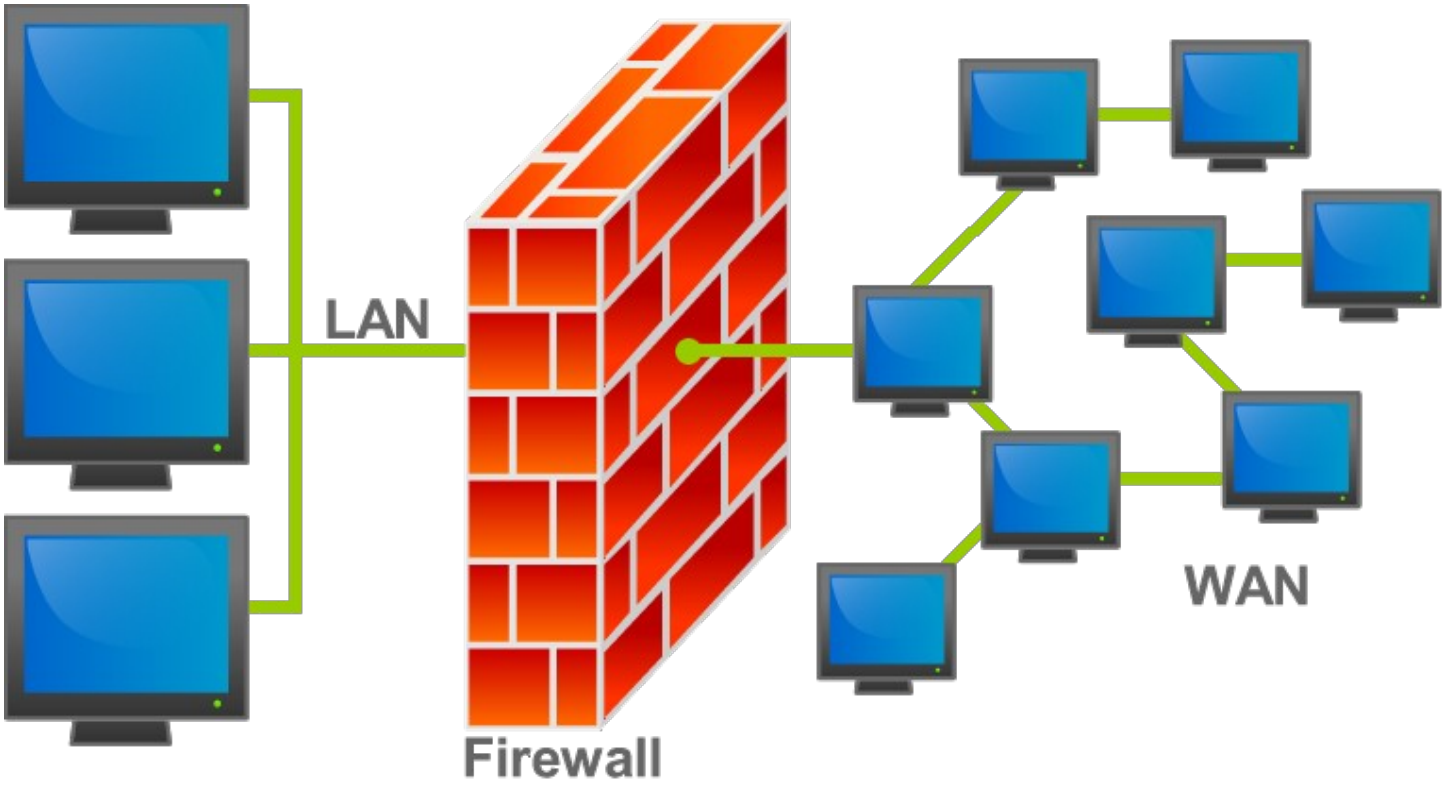


https://en.wikipedia.org/wiki/Network_address_translation#/media/File:NAT_Concept-en.svg



https://en.wikipedia.org/wiki/Carrier-grade_NAT#/media/File:CGN_IPv4.svg

Firewall



https://en.wikipedia.org/wiki/Firewall_(computing)#/media/File:Firewall.png

Stateful *vs.* stateless

- Stateful has to refer to conntrack (or something like it)

```
ACCEPT     all  --  anywhere             192.168.122.0/24     ctstate RELATED,ESTABLISH
ED
ACCEPT     all  --  192.168.122.0/24     anywhere
```

- Stateless does not need to refer to conntrack

```
ACCEPT     all  --  8.8.8.8              anywhere
```

conntrack

```
marek@mrnew:~$ sudo conntrack -L
tcp      6 431995 ESTABLISHED src=192.168.8.144 dst=85.10.202.207 sport=53370 dport=443 src=8
tcp      6 96 TIME_WAIT src=192.168.8.144 dst=216.58.201.174 sport=46610 dport=443 src=216.58
tcp      6 431996 ESTABLISHED src=192.168.8.144 dst=216.58.201.142 sport=56838 dport=443 src=
tcp      6 431993 ESTABLISHED src=192.168.8.144 dst=216.58.201.131 sport=58700 dport=443 src=
tcp      6 431977 ESTABLISHED src=192.168.8.144 dst=216.58.211.37 sport=36436 dport=443 src=2
tcp      6 431968 ESTABLISHED src=192.168.8.144 dst=216.58.201.138 sport=50936 dport=443 src=
tcp      6 431997 ESTABLISHED src=192.168.8.144 dst=64.233.184.189 sport=39562 dport=443 src=
tcp      6 431979 ESTABLISHED src=192.168.8.144 dst=172.217.168.174 sport=51622 dport=443 src
tcp      6 263 ESTABLISHED src=192.168.8.144 dst=216.58.211.37 sport=36428 dport=443 src=216.
tcp      6 431991 ESTABLISHED src=192.168.8.144 dst=172.217.168.174 sport=51570 dport=443 src
tcp      6 431996 ESTABLISHED src=192.168.8.144 dst=147.135.78.157 sport=39234 dport=443 src=
tcp      6 273 ESTABLISHED src=192.168.8.144 dst=172.217.17.10 sport=46478 dport=443 src=172.
tcp      6 431996 ESTABLISHED src=192.168.8.144 dst=216.58.201.131 sport=59140 dport=443 src=
tcp      6 431993 ESTABLISHED src=192.168.8.144 dst=52.44.211.134 sport=42430 dport=443 src=5
tcp      6 291 ESTABLISHED src=192.168.8.144 dst=172.217.16.238 sport=52550 dport=443 src=172
tcp      6 299 ESTABLISHED src=192.168.8.144 dst=74.125.140.189 sport=43698 dport=443 src=74.
tcp      6 263 ESTABLISHED src=192.168.8.144 dst=74.125.140.188 sport=43592 dport=5228 src=74
conntrack v1.4.4 (conntrack-tools): 17 flow entries have been shown.
```

https://blog.cloudflare.com/conntrack-tales-one-thousand-and-one-flows/

The netfilter.org project

What is the netfilter.org project?

The netfilter project is a community-driven collaborative FOSS project that provides packet filtering software for the Linux 2.4.x and later kernel series. The netfilter project is commonly associated with iptables and its successor nftables.

The netfilter project enables packet filtering, network address [and port] translation (NA[P]T), packet logging, userspace packet queueing and other packet mangling.

The netfilter hooks are a framework inside the Linux kernel that allows kernel modules to register callback functions at different locations of the Linux network stack. The registered callback function is then called back for every packet that traverses the respective hook within the Linux network stack.

iptables is a generic firewalling software that allows you to define rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).

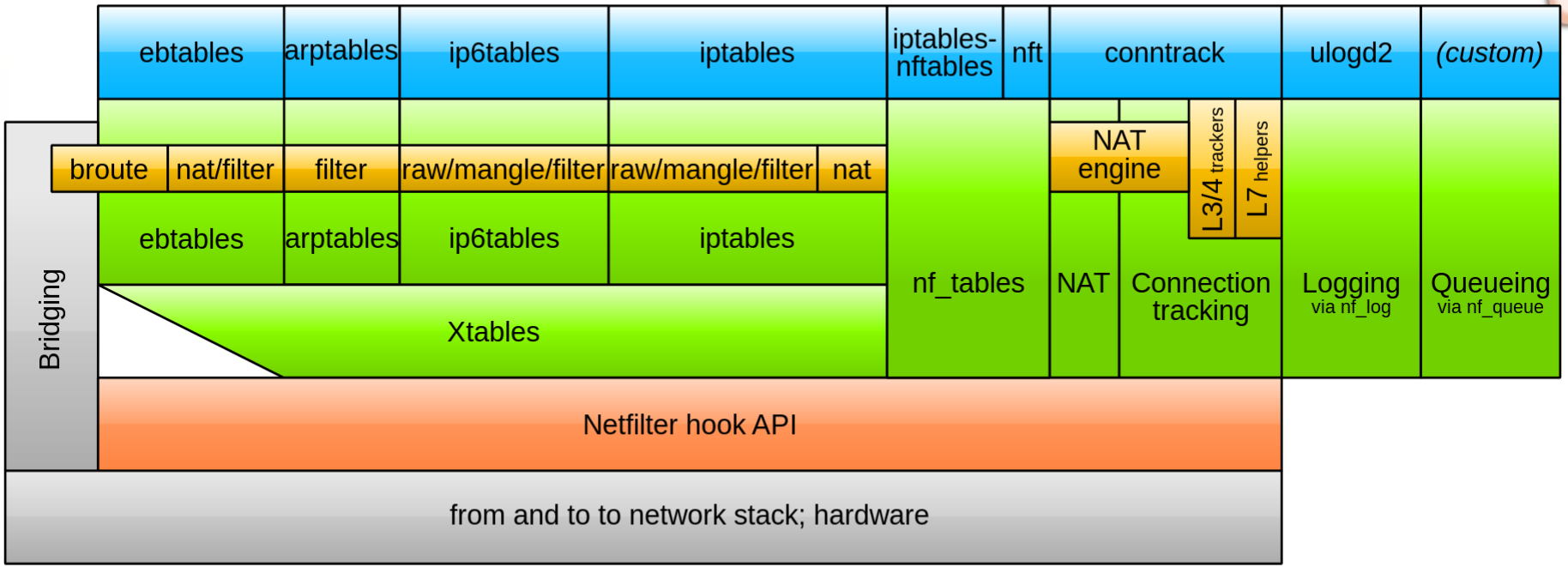
nftables is the successor of iptables, it allows for much more flexible, scalable and performance packet classification. This is where all the fancy new features are developed.

Alternatives to Netfilter

- FreeBSD
  - IPFW, natd, IPFILTER, PF
- Russia's TSPU
  - Appears to be a custom implementation and not one of the above
- SOCKS proxies in user space, instead of NAT
  - Used by Tor, ShadowSocks, etc.
- Great Cannon? Great Firewall?
  - I don't know one way or the other (maybe user space stack, or based on Linux, combination of both… who knows?)

Netfilter components

Jan Engelhardt, last updated 2014-02-28 (initial: 2008-06-17)



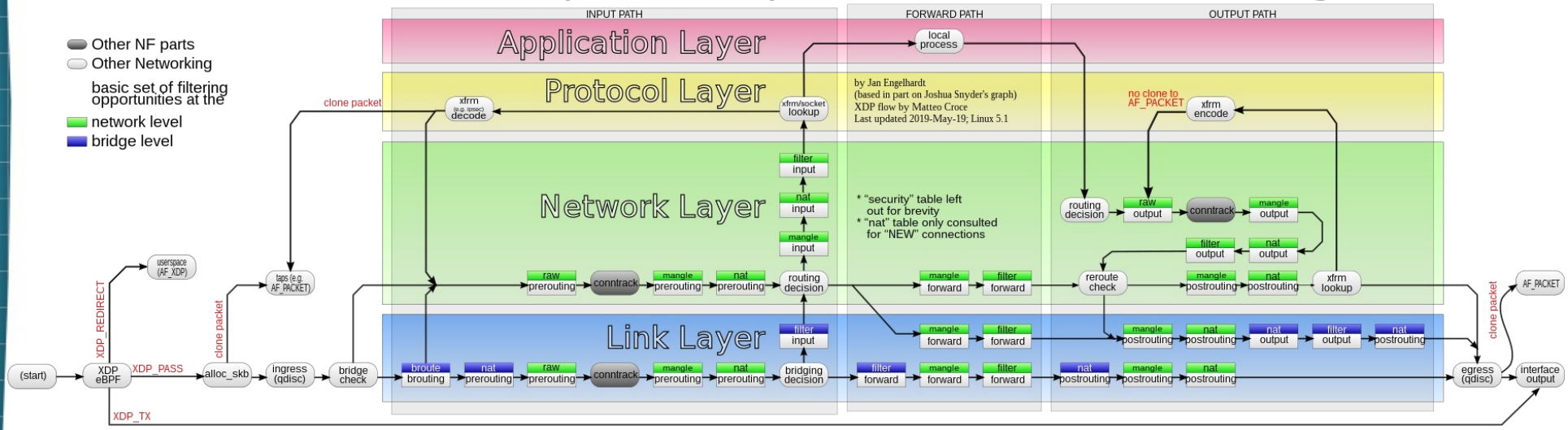
Userspace tools

Netfilter kernel components

other networking components

https://en.wikipedia.org/wiki/Netfilter

Packet flow in Netfilter and General Networking

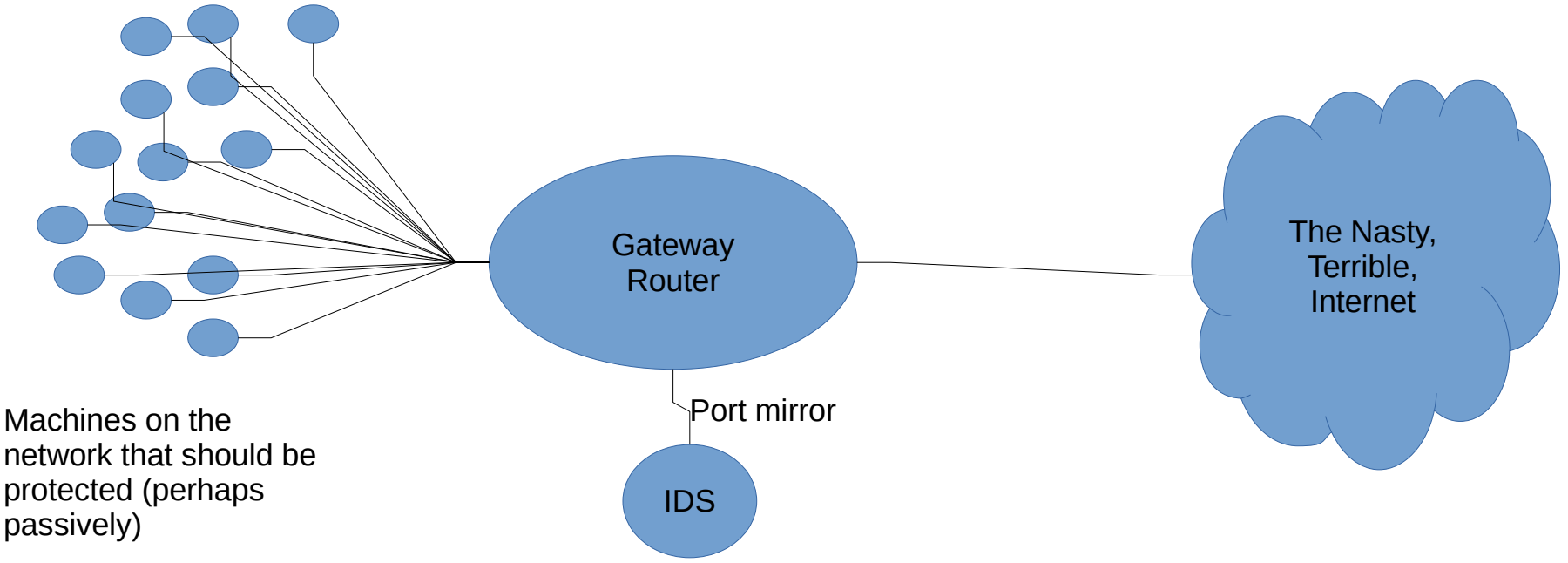


https://en.wikipedia.org/wiki/Netfilter

Network Intrusion Detection System

- NIDS
  - Can be in-path or on-path
  - Can be passive or active
    - Log a report, inject RSTs, drop, ...
  - Anomoly-based *vs.* rule-based
  - Sometimes the line between firewall and NIDS is not clear
    - Typically firewalls operate in layers 3 and 4 and are in-path, typically NIDS operates in layers 3 through 7 and are on-path

Intrusion Detection System (examples are Bro or Snort)



Information controls

- Machine-in-the-middle
  - Great Cannon is an example (in-path)
- Machine-on-the-side
  - Great Firewall of China (GFW, on-path) and NSA QUANTUM are examples
- TTL is a clue, but is easy to hide

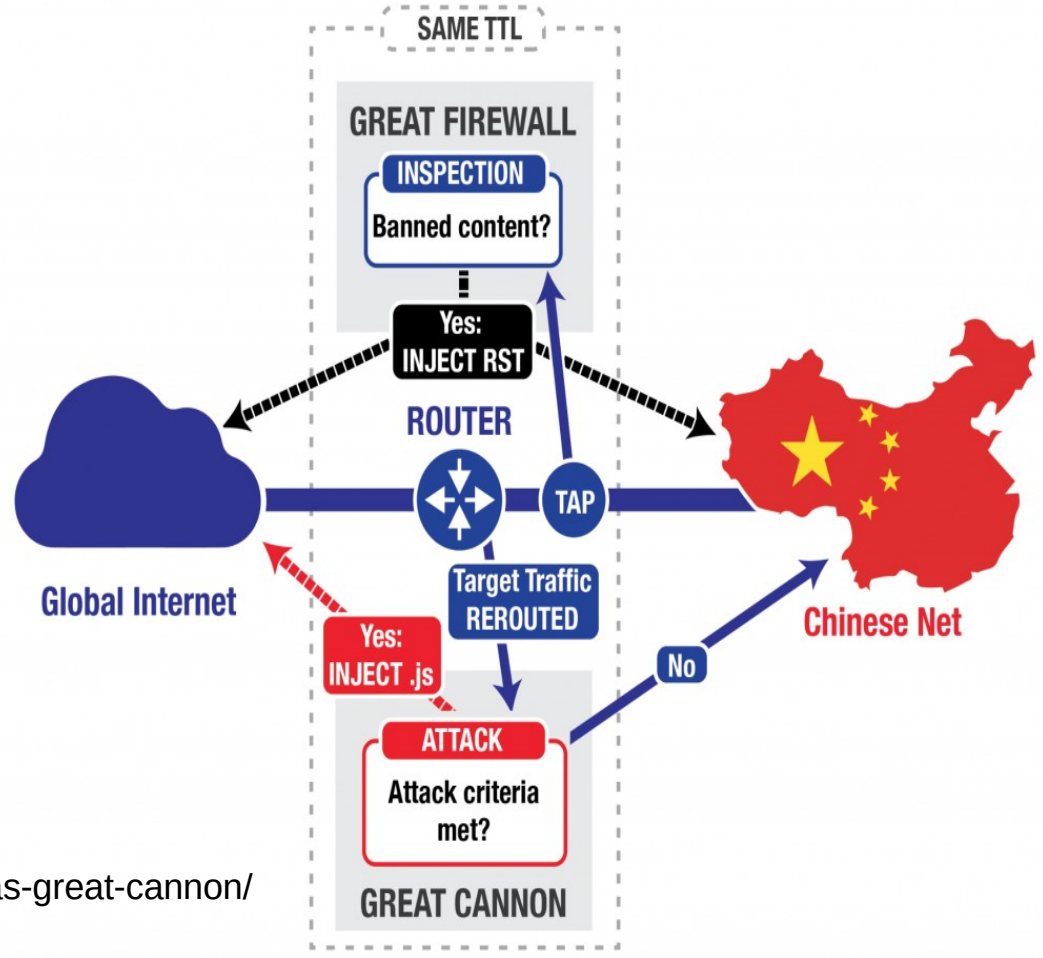


Image reproduced from https://citizenlab.org/2015/04/chinas-great-cannon/

Snort rule examples from https://cyvatar.ai/write-configure-snort-rules/ ...

Case 1: Securing Email Server With Snort Rules:

alert tcp 192.168.1.0/24 any -> 131.171.127.1 25 (content: “hacking”; msg: ”malicious packet”; sid:2000001;)

Case 2: Detecting TCP SYN Floods

Alert tcp any any -> 192.168.10.5 443 (msg: “TCP SYN flood”; flags:!A; flow: stateless; detection_filter: track by_dst, count 70, seconds 10; sid:2000003;)

Case 3: Securing your Network against Conficker A Worm

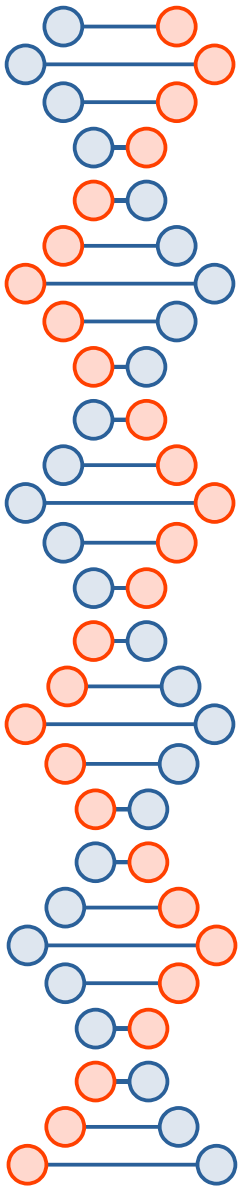
alert tcp any any -> any 445 (msg: “conficker.a shellcode”; content: “|e8 ff ff ff ff c1|^|8d|N|10 80|1|c4|Af|81|9EPu|f5 ae c6 9d a0|O|85 ea|O|84 c8|O|84 d8|O|c4|O|9c cc|IrX|c4 c4 c4|,|ed c4 c4 c4 94|&<O8|92|\;|d3|WG|02 c3|,|dc c4 c4 c4 f7 16 96 96|O|08 a2 03 c5 bc ea 95|\;|b3 c0 96 96 95 92 96|\;|f3|\;|24|i| 95 92|QO|8f f8|O|88 cf bc c7 0f f7|2I|d0|w|c7 95 e4|O|d6 c7 17 f7 04 05 04 c3 f6 c6 86|D|fe c4 b1|1|ff 01 b0 c2 82 ff b5 dc b6 1b|O|95 e0 c7 1 cb|s|d0 b6|O|85 d8 c7 07|O|c0|T|c7 07 9a 9d 07 a4|fN|b2 e2|Dh|0c b1 b6 a8 a9 ab aa c4|]|e7 99 1d ac b0 b0 b4 fe eb eb|”; sid: 2000002; rev: 1;)

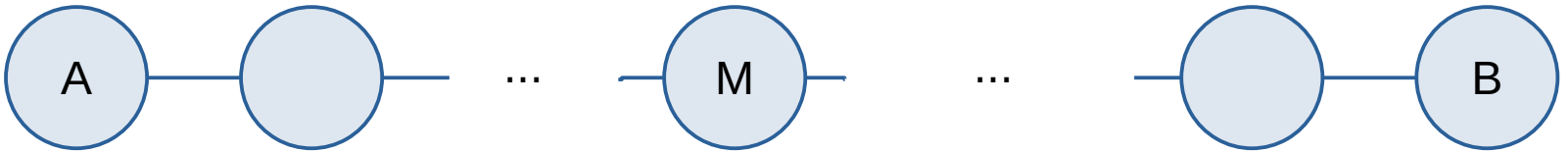
Case 4: Alerts of Buffer Overflow in BIND

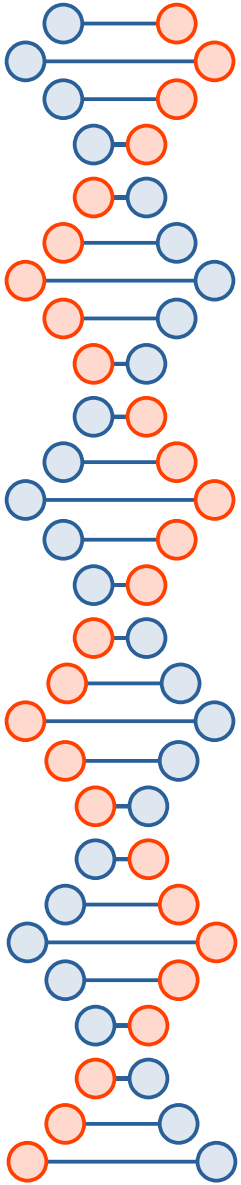
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:”FTP wuftp bad file completion attempt [“;flow:to_server, established; content:”|?|”; content:”[“; distance:1; reference:bugtraq,3581; reference:bugtraq,3707; reference:cve,2001-0550; reference:cve,2001-0886; classtype:misc-attack; sid:1377; rev:14;)

Firewalls and Deep Packet Inspection (DPI)

- Dual use technology
  - Network access controls
  - Security monitoring and response
  - Load balancing
  - NAT
  - VPNs
  - Surveillance (“userid=*, longlat=*”)
  - Censorship (“falun”)
  - Throttling (“twitter.com”)
  - Targeted attacks (“HTTP GET cbjs.baidu.com/js/o.js”)

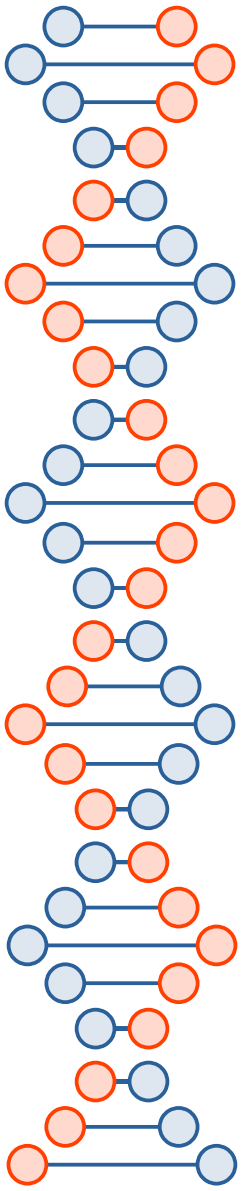



A
...
M
...
B



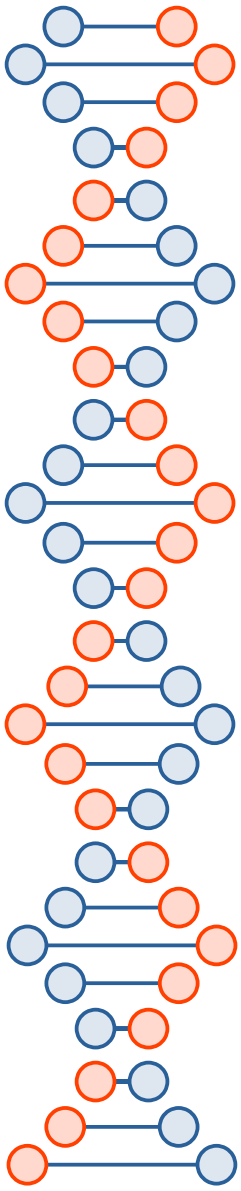
Basic setup...

- A and B are communicating by sending packets back and forth
  - Packets can be dropped, re-ordered
- Packets are routed by routers, including M (the middlebox)
  - *I.e.*, M is “in-path”
- M wants to know the contents of their messages
- Ignore encryption for now
  - *E.g.*, middlebox wants to see SNI



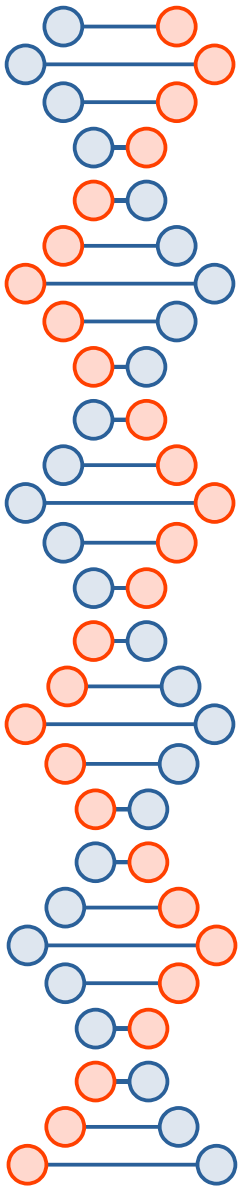
Basic questions

- Can M know the state of the connection?
  - *E.g.*, is the connection between A and B a new one? An old one? Is there a valid connection at all?
- Can M know exactly what A said to B, or *vice versa*?



Short answer: no

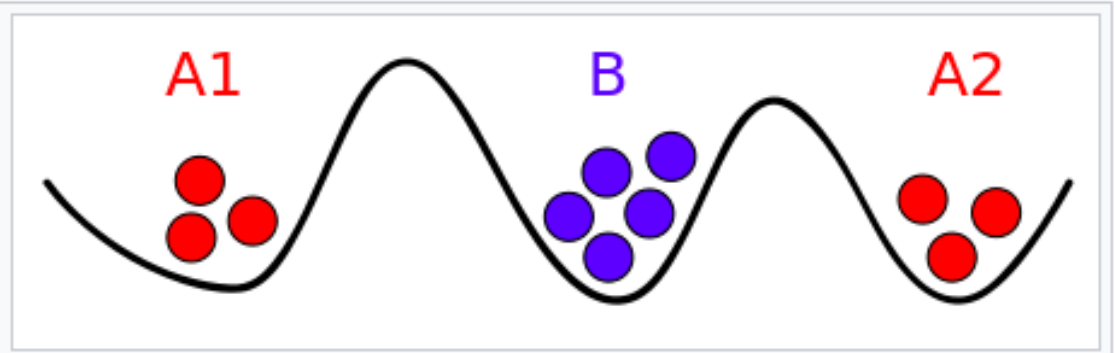
- A, B, and M may implement the protocol differently
  - Postel's law: be conservative in what you do, be liberal in what you accept from others
- It gets worse… The Internet is designed end-to-end, there is a lot of state in A and B that M is not privy to
  - "Correctness" is defined as a partial ordering in distributed systems
- It gets worse… Even if they all agreed on the protocol, it's impossible even for A and B to know the state of the connection they share unambiguously



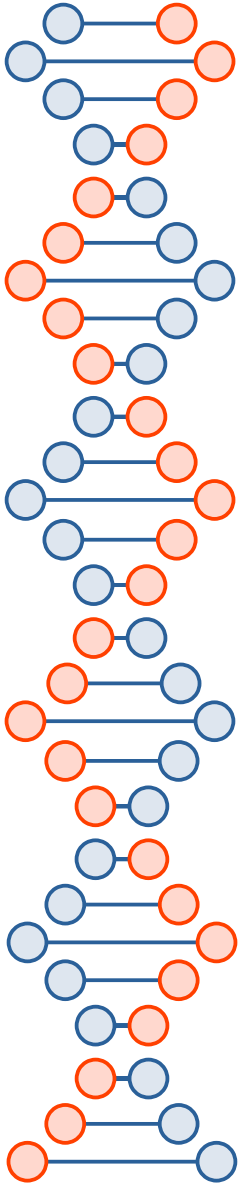
In reverse order...

Two Generals' Problem

- https://en.wikipedia.org/wiki/Two_Generals%27_Problem

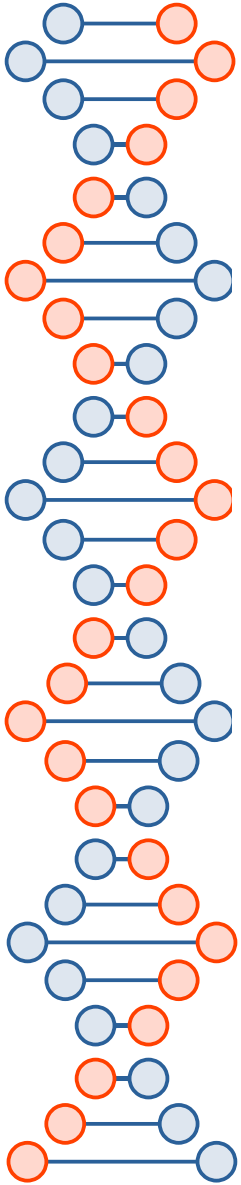


Positions of the armies. Armies A1 and A2 cannot see one another directly, so need to communicate by messengers, but their messengers may be captured by army B.



Two Generals' Problem

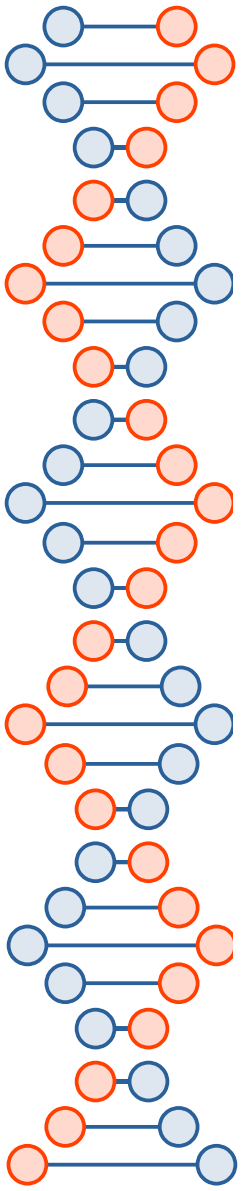
- A1 and A2 need to agree on when to attack
  - Same as A and B in our example unambiguously knowing the state of their own connection (open, half-open, closed?), and other state about the connection



SOME CONSTRAINTS AND TRADEOFFS IN THE DESIGN OF NETWORK COMMUNICATIONS*

E. A. Akkoyunlu
K. Ekanadham
R. V. Huber†
Department of Computer Science
State University of New York at Stony Brook

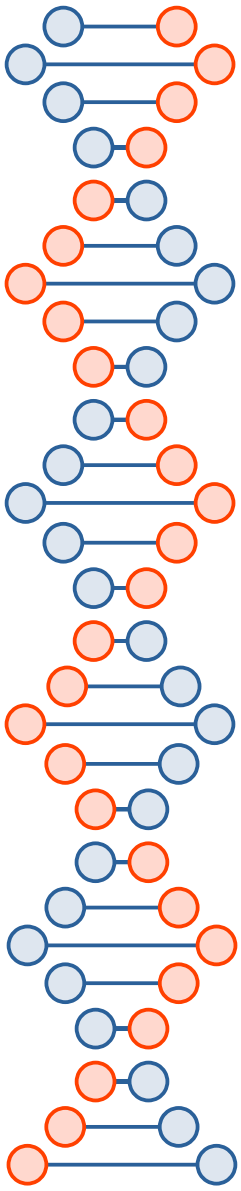
SOSP '75: Proceedings of the fifth ACM symposium on Operating systems principles • November 1975 • Pages 67–74
• https://doi.org/10.1145/800213.806523



APPENDIX: User Implemented Protocols

To show that no amount of user protocol can solve the problem in a manner to dissipate the anxiety of both parties as to the outcome of a transaction, consider the following model.

A group of gangsters are about to pull off a big job. The plan of action is prepared down to the last detail. Some of the men are holed up in a warehouse across town, awaiting precise instructions. It is absolutely essential that the two groups act with complete reliance on each other in executing the plan.



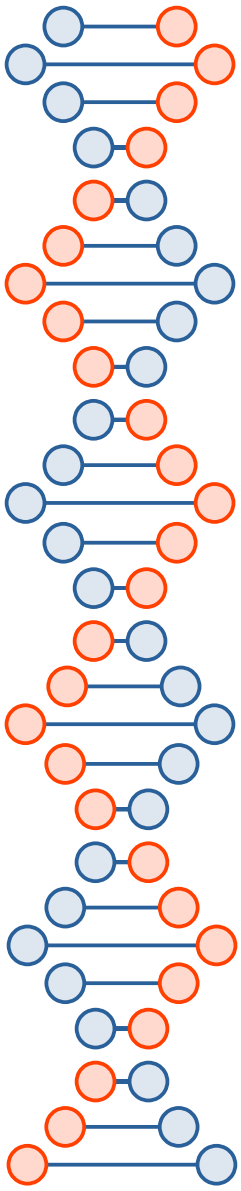
Of course, they will never get around to putting the plan into action, because the following sequence of events is bound to take place.

1. A messenger is dispatched across town, with instructions from the boss.
2. The messenger reaches his destination. At this point both parties know the plan of action. But the boss doesn't know that his message got through (muggings are a common occurrence). So the messenger is sent back, to confirm the message.

3. The messenger reaches the boss safely. Now, everybody knows the message got through. Of course, the men in the warehouse are not aware that step 3 occurred, and must be reassured. Off goes the messenger.
4. Now the men in the warehouse too know that step 3 was successful, but unless they communicate their awareness...

.....

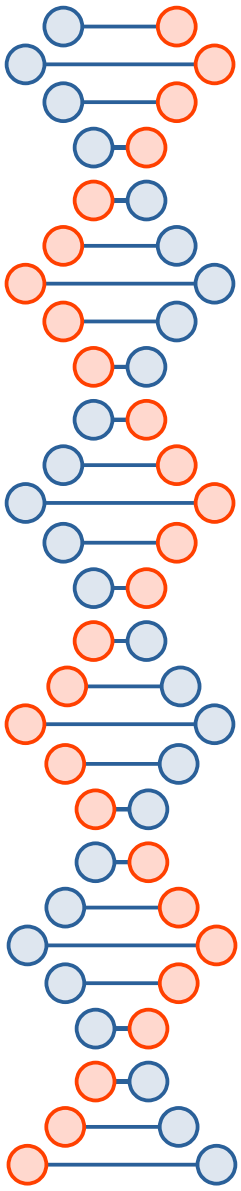
.....



Note that the needs of both parties are quite reasonable. They simply want to reach a state where

(1) The original message (i.e., the plan of action) is successfully delivered, and

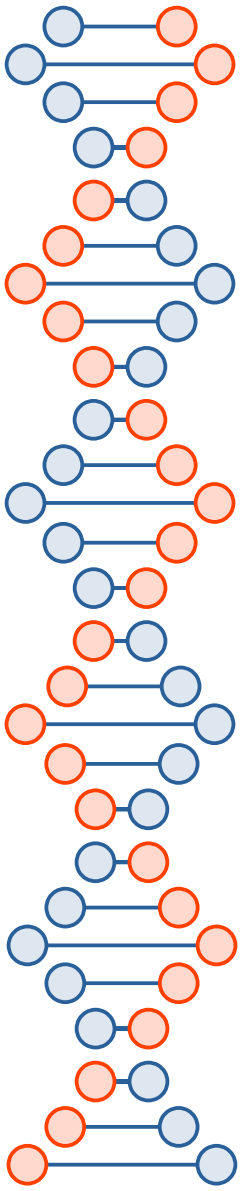
(2) Both parties know that they are in mutual agreement that (1) occurred.



Fact The sequence cannot terminate successfully.

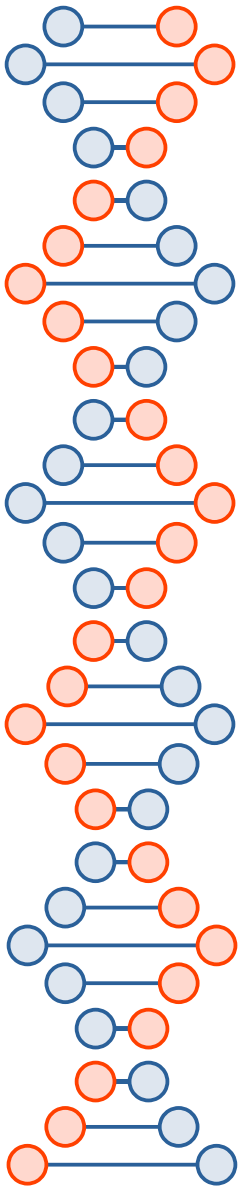
Proof (a) Clearly the sequence contains at least one message of importance.

(b) Assume that it is possible to reach the desired state after a finite sequence of messages. Then there must exist a number $n \geq 1$ such that n is the length of the shortest sequence which gets us to this state. Since this is the shortest sequence, the last message in it is important: if the n'th message gets lost, the desired state cannot be reached. The sender of the n'th message must receive acknowledgment. This means that the sequence is at least of length $n + 1$. The assumption is contradicted and the sequence cannot be finite.



Note also that the sequence is infinite even when none of the messages are actually lost.

At first glance it would seem that if the two processes are in continuous communication, the problem can be solved by including a sequence number [8] as part of each message. But this is not really so: sequence numbers are analogous to the step numbers in the above example. At any time the process receiving the highest numbered message knows the complete state while the other lives in doubt. Thus in practice only sequential events can be controlled but simultaneity cannot be achieved by this means.



Even if they all agreed on the protocol, it’s impossible even for A and B to know the state of the connection they share unambiguously

(good news if you’re trying to evade censorship, bad news if you’re trying to keep malware off your network)

TCP 3-way handshake (review)

- TCP header has flags
  - SYN is “Synchronize”, it means the sequence number has a special meaning
  - ACK is “Acknowledge”, it means the acknowledgment number has meaning
  - RST: “I have no record of such a connection”
  - Also, FIN, CWR, ECN, URG, PUSH

TCP 3-way handshake (review)

- SYN: I'd like to open a connection with you, here's my initial sequence number (ISN)
- SYN/ACK: Okay, I acknowledge your ISN and here's mine
- I ACK your ISN

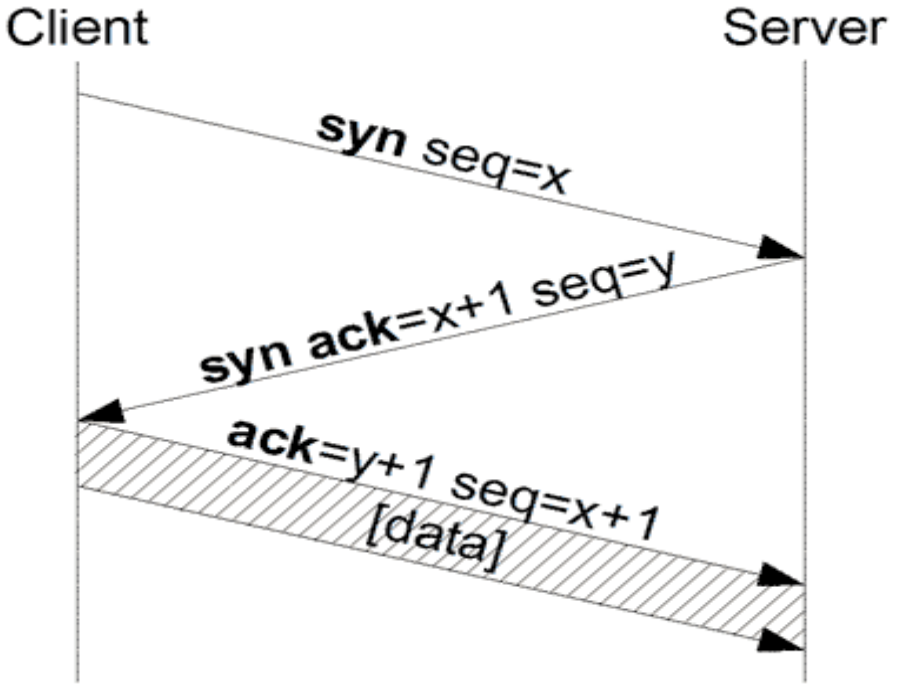


Image from Wikipedia

NIDS evasion example: TTL limiting

- Victim is 10 hops away from you (the attacker)
- IDS is 7 hops away from you, 3 from the victim
- Send a SYN with TTL 64
- Get a SYN/ACK from the victim
- Send a RST with TTL 9
- Send an ACK with TTL 64
- Victim sees SYN, sends SYN/ACK, gets ACK, you have an open connection and can send them data
- IDS sees SYN in one direction, SYN/ACK in the other, then RST and assumes the connection was reset, ACK and all packets that follow (with data) are ignored by the IDS

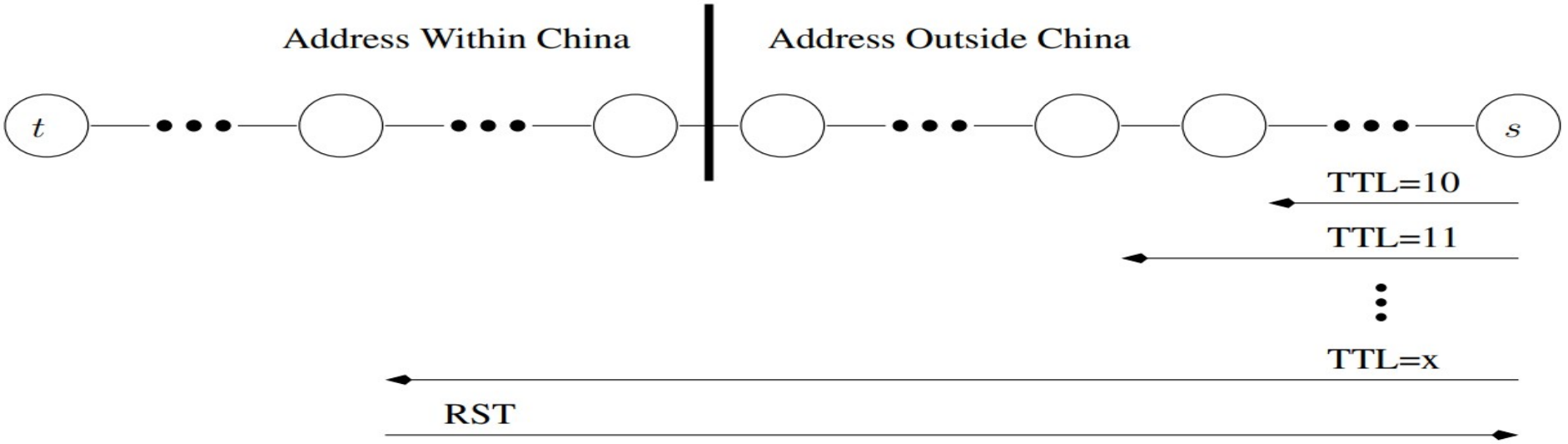
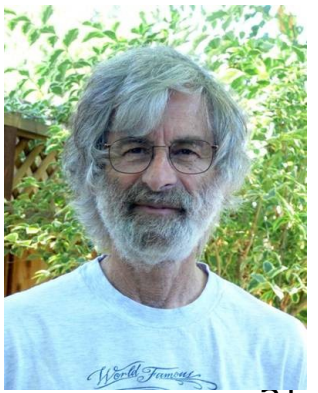


Figure 4: GFC router discovery using TTLs.

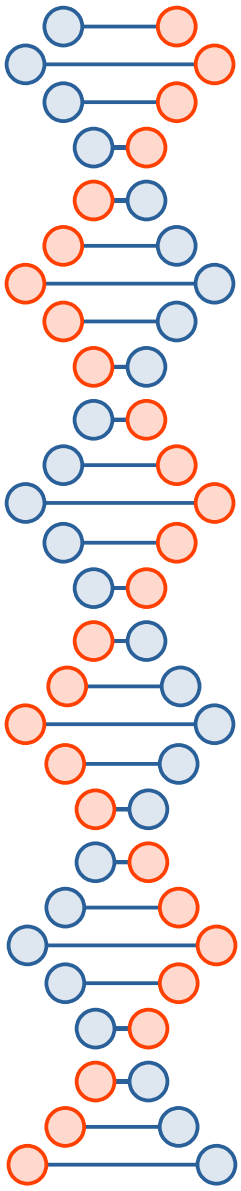
Reproduced from:
https://jedcrandall.github.io/concept_doppler_ccs07.pdf

Leslie Lamport

- Microsoft Research
- “winner of the 2013 Turing Award for imposing clear, well-defined coherence on the seemingly chaotic behavior of distributed computing systems”
- Also… LaTeX, Lamport signatures, temporal logic, ...
- https://en.wikipedia.org/wiki/Leslie_Lamport



http://lamport.org/



Communications of the ACM, Volume 21, Issue 7 • July 1978

Operating Systems

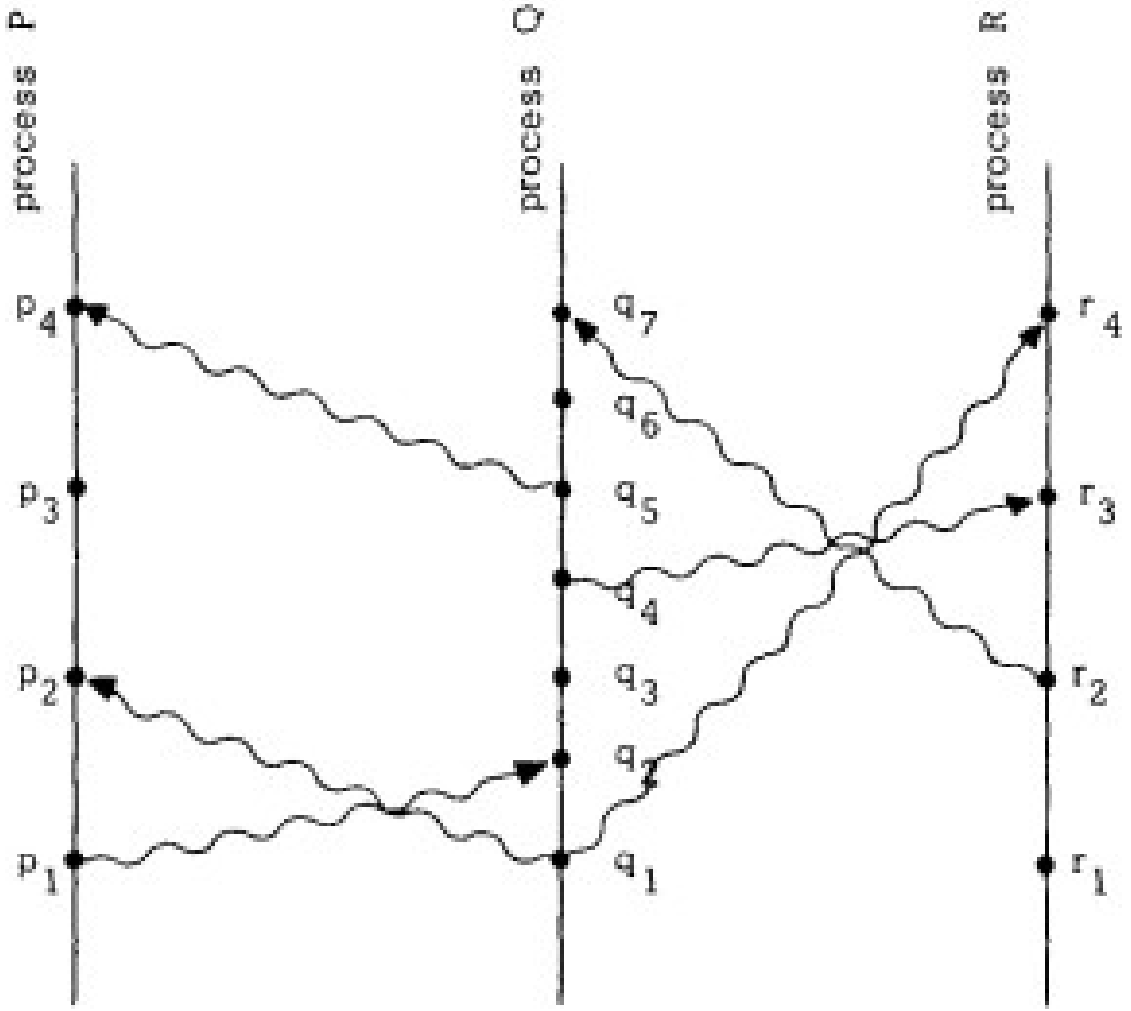
R. Stockton Gaines Editor

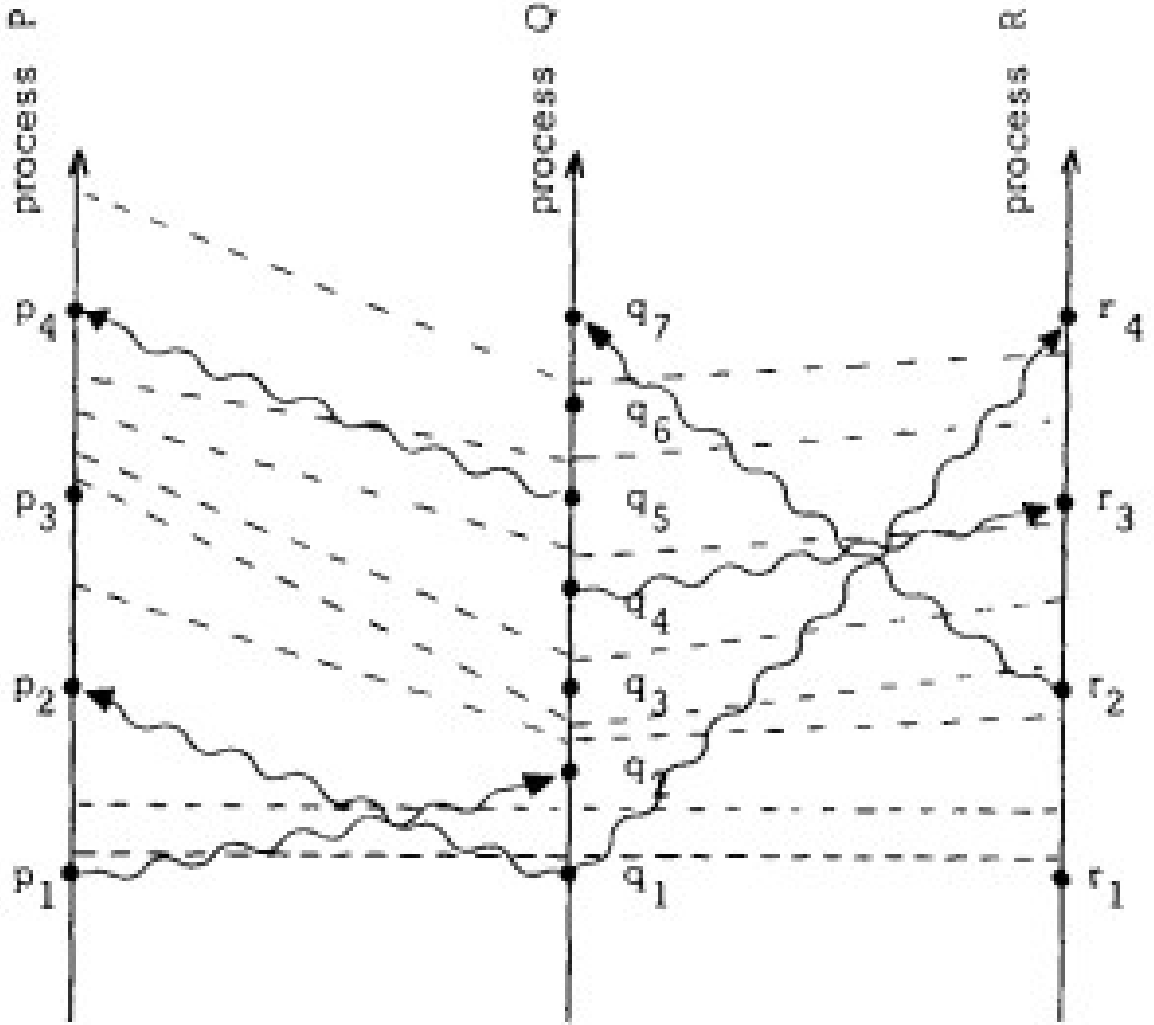
Time, Clocks, and the Ordering of Events in a Distributed System

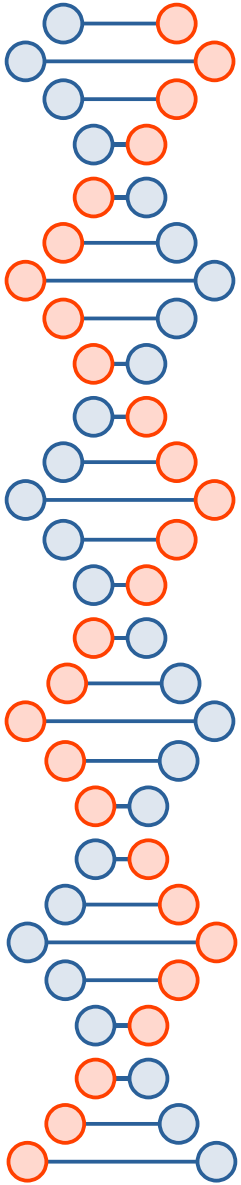
Leslie Lamport
Massachusetts Computer Associates, Inc.

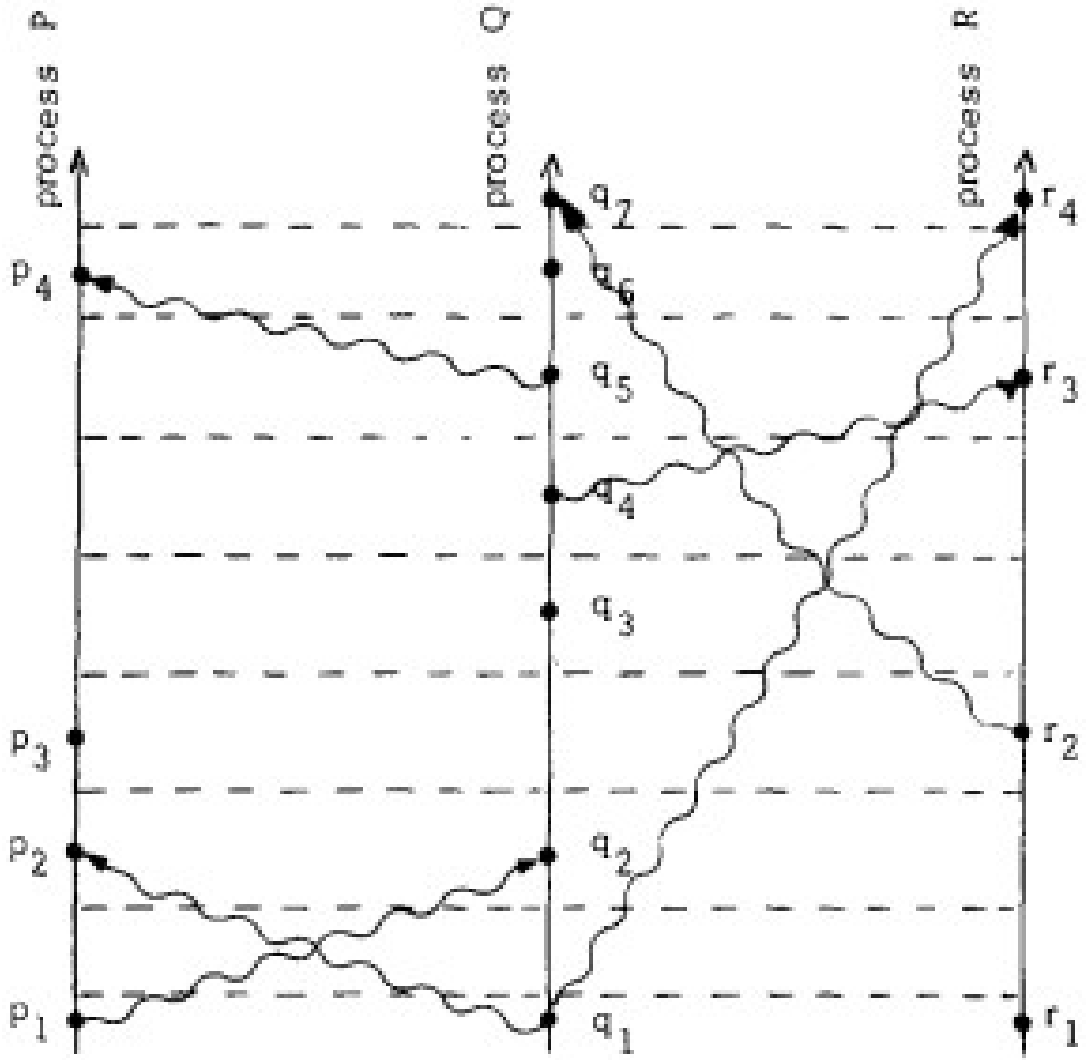
The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

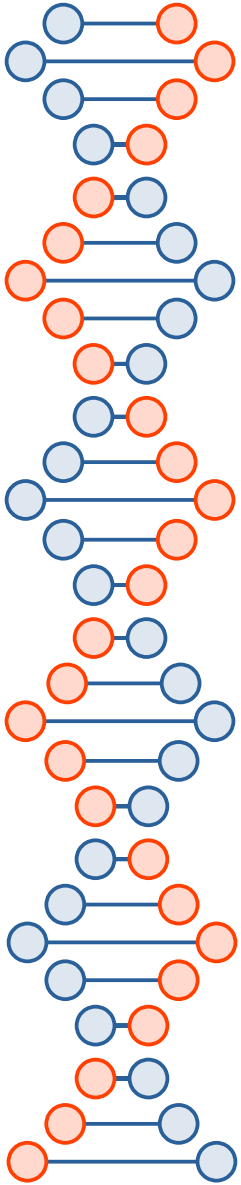
Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems


process P
process Q
process R
p_4
p_3
p_2
p_1
q_7
q_6
q_5
q_4
q_3
q_2
q_1
r_4
r_3
r_2
r_1

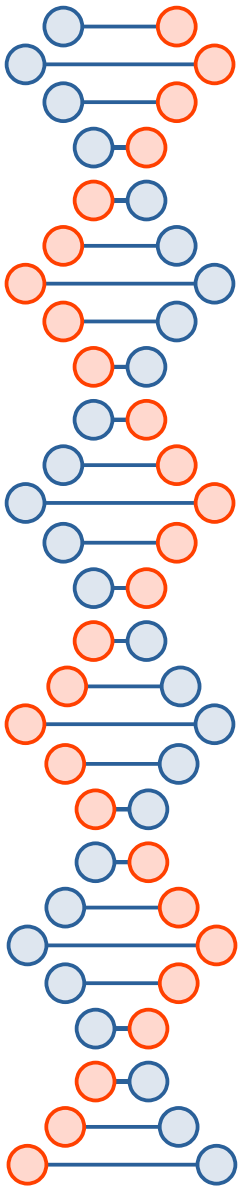

P
process
Q
process
R
process
p_4
p_3
p_2
p_1
q_7
q_6
q_5
q_4
q_3
q_2
q_1
r_4
r_3
r_2
r_1



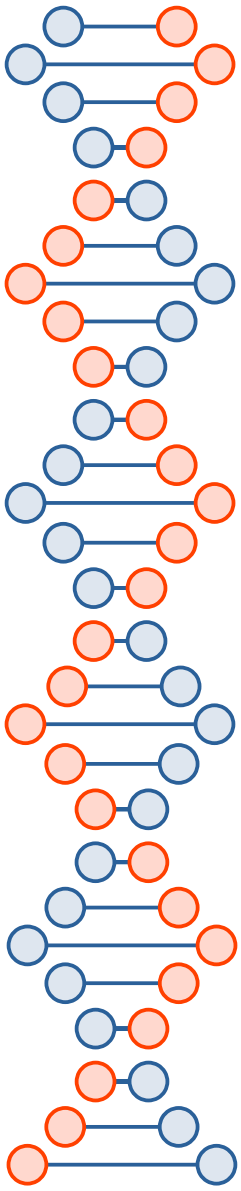

process P
process Q
process R
p_4
p_3
p_2
p_1
q_2
q_5
q_4
q_3
q_2
q_1
r_4
r_3
r_2
r_1



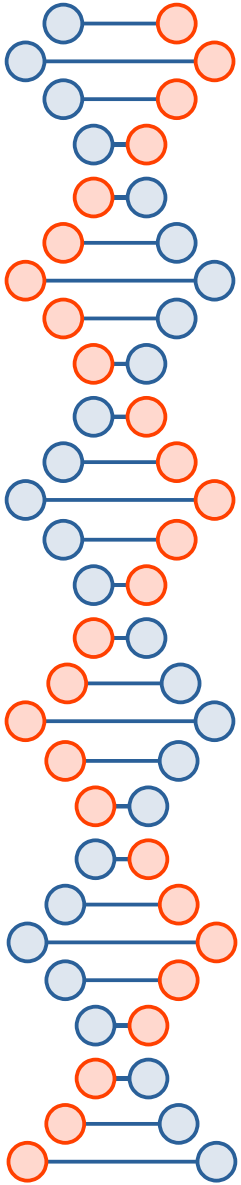
In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation "happened before" is therefore only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.



Another way of viewing the definition is to say that $a \rightarrow b$ means that it is possible for event a to causally affect event b. Two events are concurrent if neither can causally affect the other. For example, events p_3 and q_3 of Figure 1 are concurrent. Even though we have drawn the diagram to imply that q_3 occurs at an earlier physical time than p_3, process P cannot know what process Q did at q_3 until it receives the message at p_4. (Before event p_4, P could at most know what Q was *planning* to do at q_3.)



This definition will appear quite natural to the reader familiar with the invariant space-time formulation of special relativity, as described for example in [1] or the first chapter of [2]. In relativity, the ordering of events is defined in terms of messages that *could* be sent. However, we have taken the more pragmatic approach of only considering messages that actually *are* sent. We should be able to determine if a system performed correctly by knowing only those events which *did* occur, without knowing which events *could* have occurred.



"Correctness" is defined as a partial ordering in distributed systems

A protocol must enforce ordering for correctness (Internet protocols like TCP and IP don't do this, especially not for the benefit of middleboxes)

The Internet is designed end-to-end, there is a lot of state in A and B that M is not privy to

NIDS evasion example: TSPU

- TSPU is Russia's censorship system
- We'll talk about it a lot for the rest of the semester

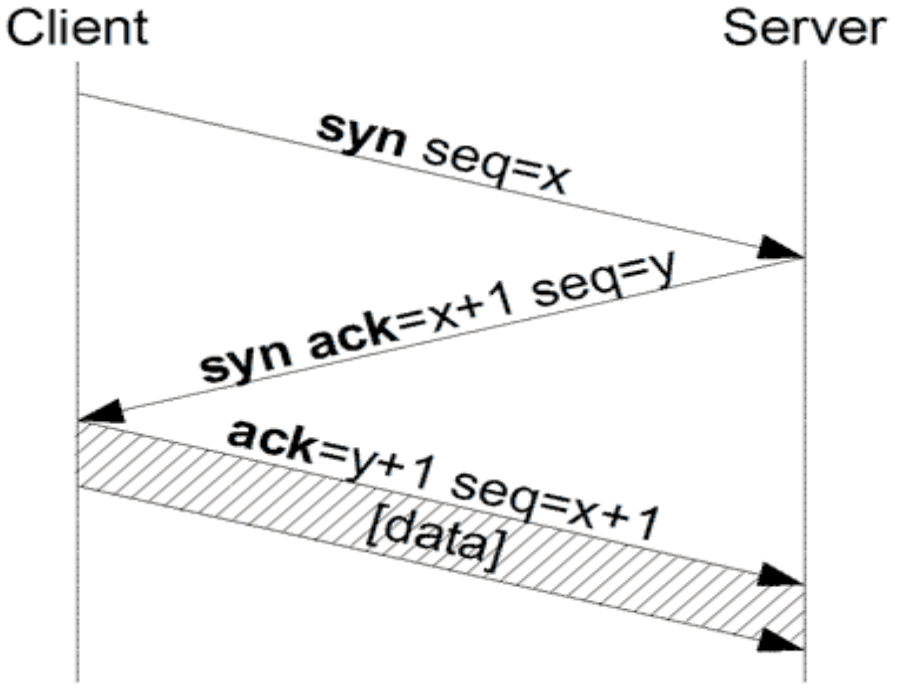
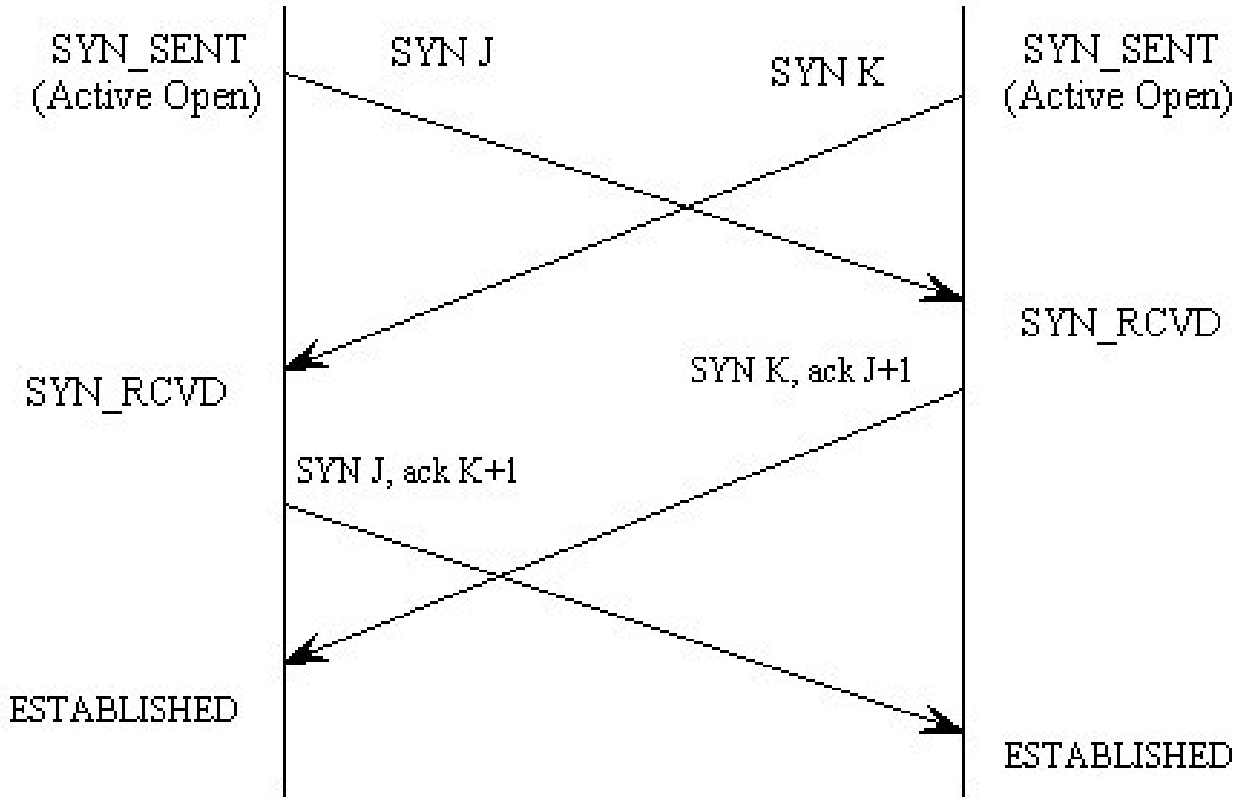


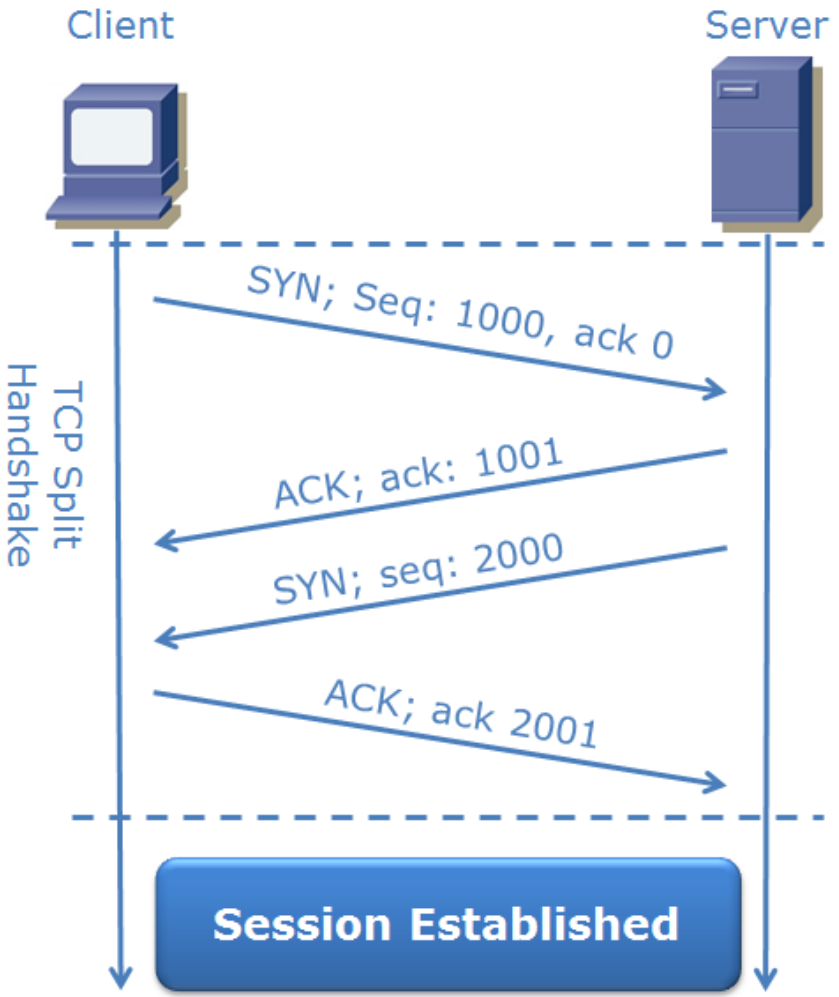
Image from Wikipedia

https://ttcplinux.sourceforge.net/documents/one/tcpstate/tcpstate.html



state transitions in simultaneous open

https://www.hackmageddon.com/2011/04/17/tcp-split-handshake-attack-explained/



https://ensa.fi/papers/tspu-imc22.pdf

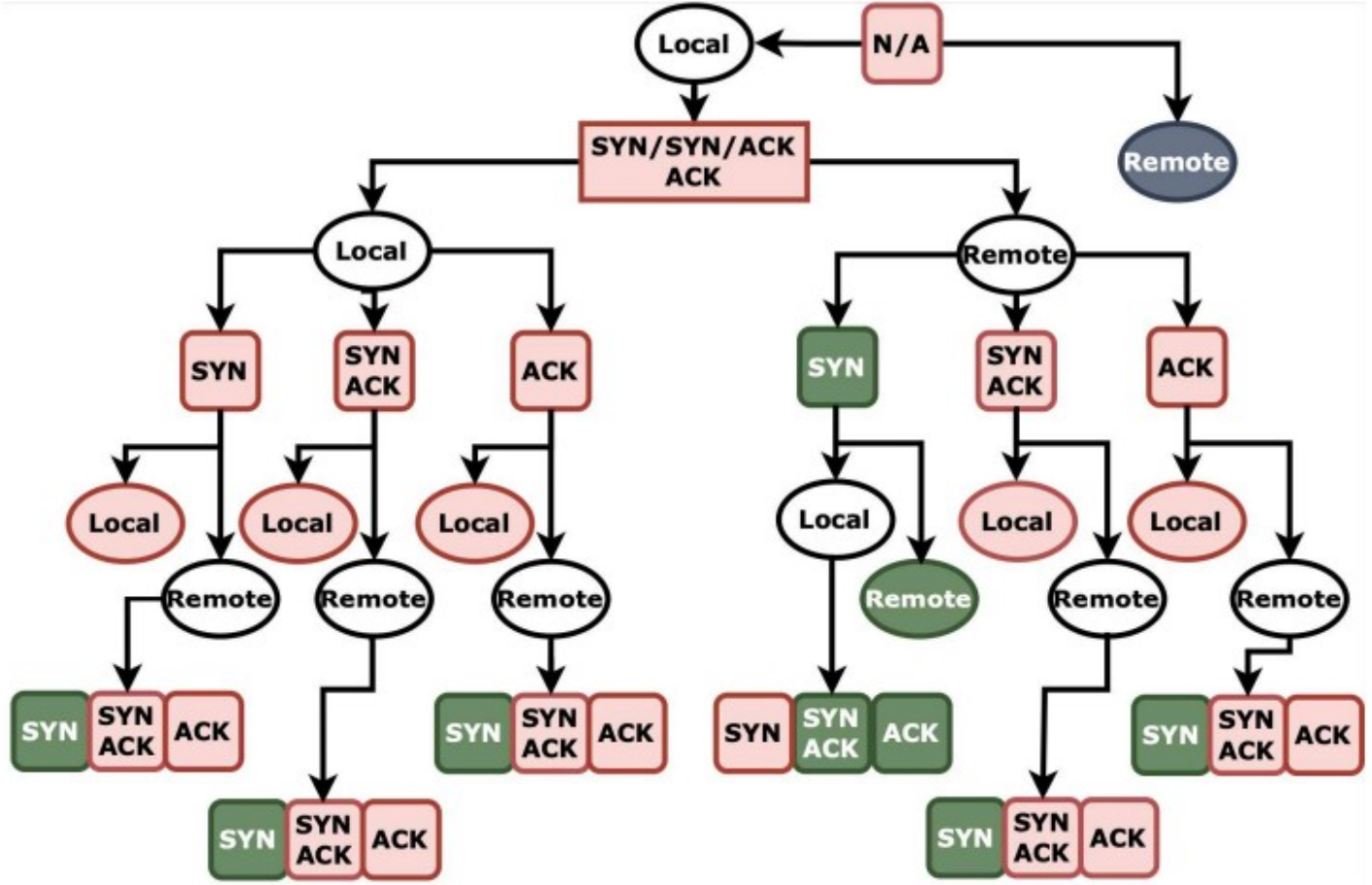
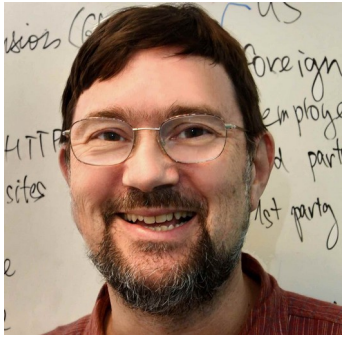


Figure 4: TSPU Triggering Sequences.

Vern Paxson



- Developer of Zeek (formerly known as Bro)
  - Laid the foundations of NIDS research
  - Concurrently discovered and published many of the same issues as Ptacek and Newsham
- Professor Emeritus of Computer Science at Berkeley and former Director of the Networking and Security Group at the International Computer Science Institute (ICSI)
- End-to-end packet dynamics on the Internet, Internet measurement, worms, ethics, Internet censorship...
- https://en.wikipedia.org/wiki/Vern_Paxson

Where do Internet standards come from?

- IETF = Internet Engineering Task Force
- RFC = Request for Comments
  - MUST, MUST NOT, SHOULD, SHOULD NOT, MAY (RFC 2119)
- “The only laws on the Internet are assembly and RFCs” -- Phrack 65
  - Assembly is an abstraction
  - RFCs are not always followed
    - Often ambiguous

IP reassembly

- Routers (or endhosts, if they want) can break IP packets up into fragments that the receiver has to reassemble
- Ambiguity in the way overlapping IP fragments are put back together into an IP packet
- All of the following images were plagiarized from:

  https://www.sans.org/reading-room/whitepapers/detection/ip-fragment-reassembly-scapy-33969

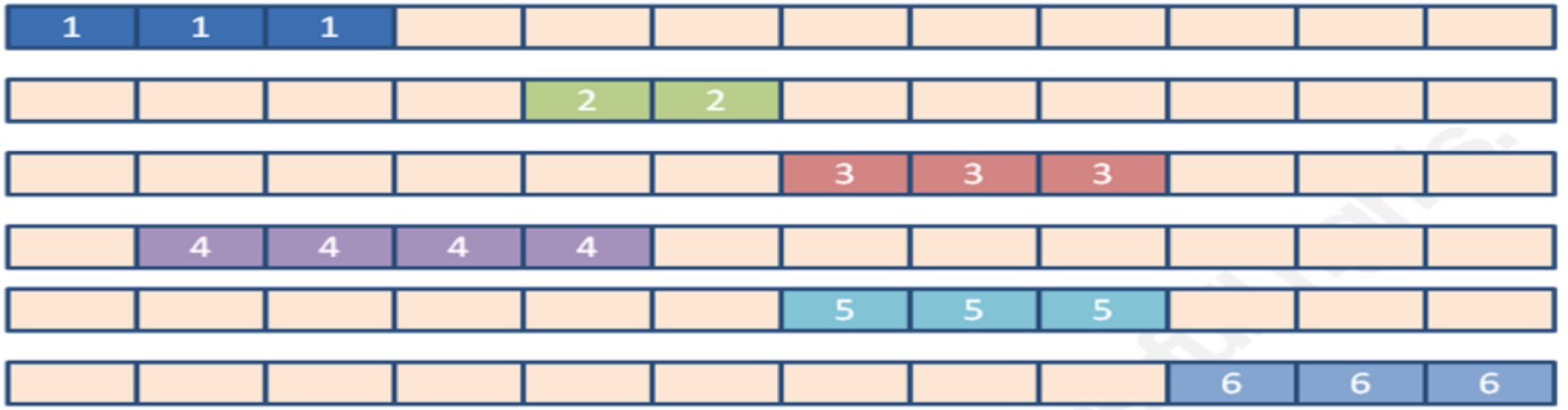


Figure 1: 6 Fragmented Packets (Shankar & Paxson, 2003)(Novak, 2005)

Reassembled using policy: First (Windows, SUN, MacOS, HPUX)

Reassembled using policy: Last/RFC791 (Cisco)

Reassembled using policy: Linux (Linux)

Reassembled using policy: BSD (AIX, FreeBSD, HPUX, VMS)

Reassembled using policy: BSD-Right (HP Jet Direct)

Figure 2: 5 Reassembly Methods (Shankar & Paxson, 2003)(Novak, 2005)

Figure 3: Views of the attacker, IDS and analyst

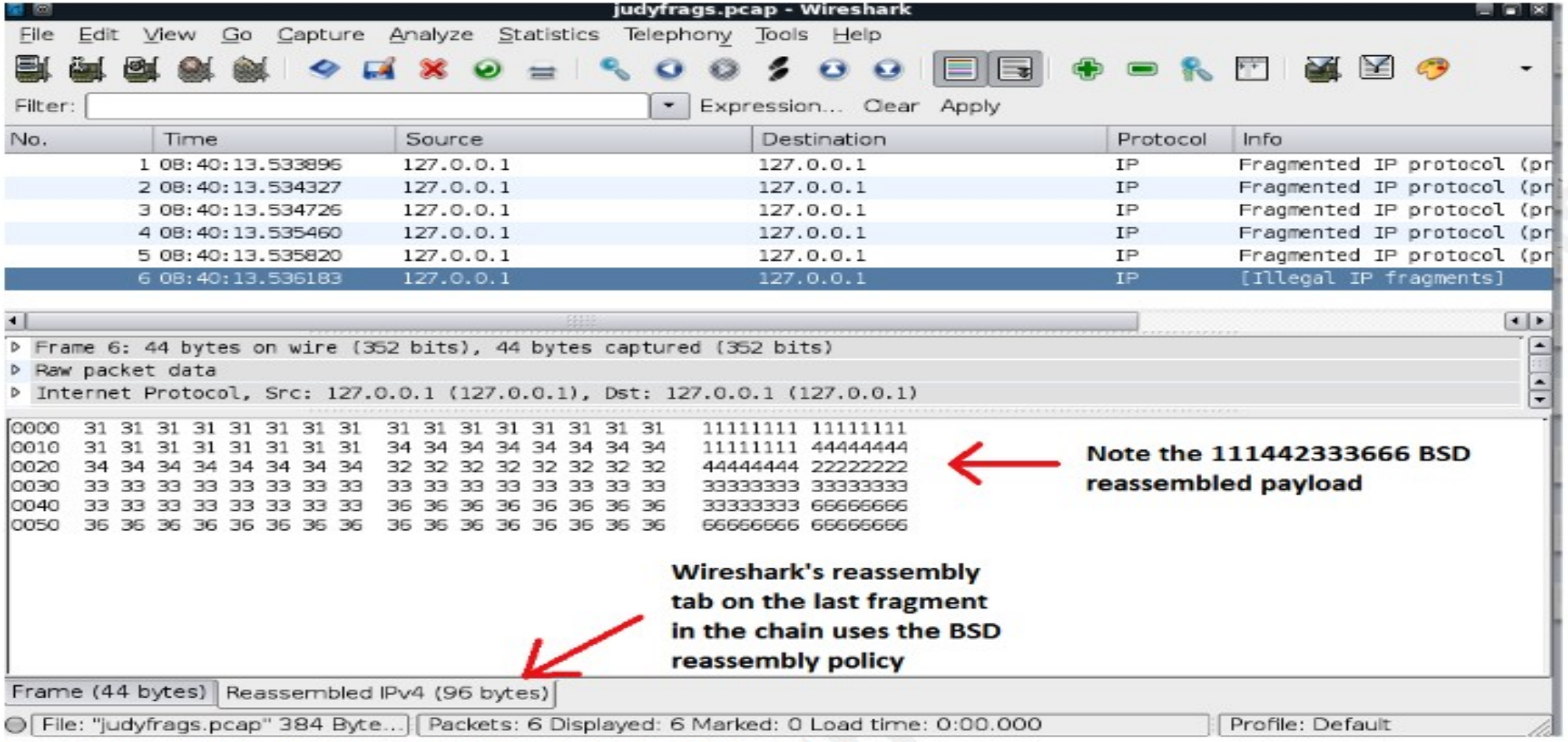
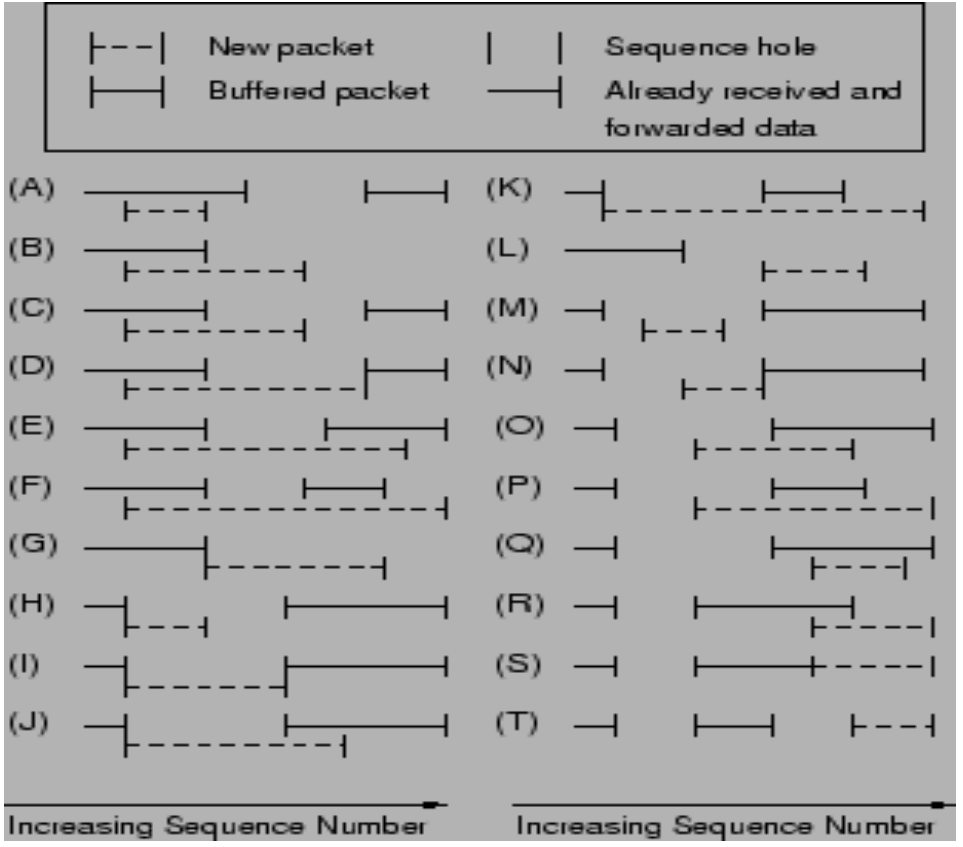


Figure 4: Wireshark uses BSD reassembly technique

TCP is even worse...

- From http://www.icir.org/vern/papers/TcpReassembly/



IDS is looking for signatures

- Typically regular expressions, like “`.*<script>.*</script>.*`” appearing in an input to a web form, indicating a Javascript XSS attack.
- How can we (the attacker) get the IDS to see one thing and the victim to see another?
- A stupid example: Great Firewall of China censors “`GET falungong.html`”, but if you send two packets: “`GET fa`” and “`lungong.html`” the endhost reassembles them fine but the GFW is fooled.
- Or, “GET fa%61lungong.html”

Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection

Thomas H. Ptacek
tqbf@securenetworks.com

Timothy N. Newsham
newsham@securenetworks.com

Secure Networks, Inc.

January, 1998

https://users.ece.cmu.edu/~adrian/731-sp04/readings/Ptacek-Newsham-ids98.pdf

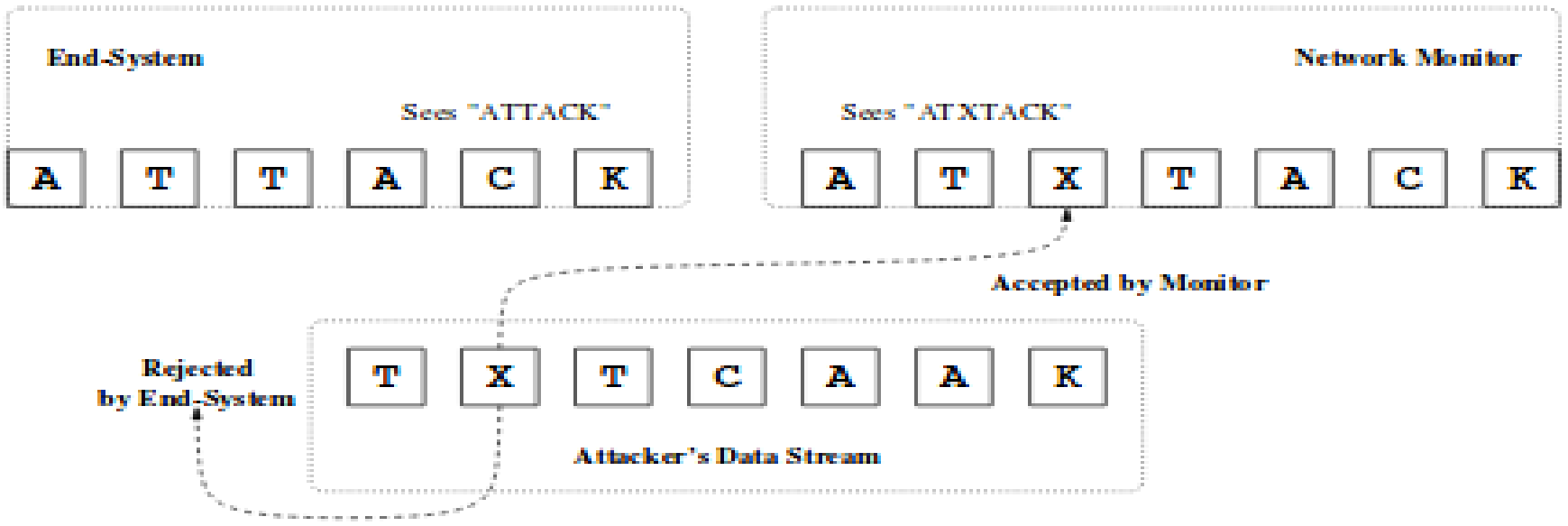


Figure 4: Insertion of the letter 'X'

Figure from Ptacek and Newsham

https://users.ece.cmu.edu/~adrian/731-sp04/readings/Ptacek-Newsham-ids98.pdf

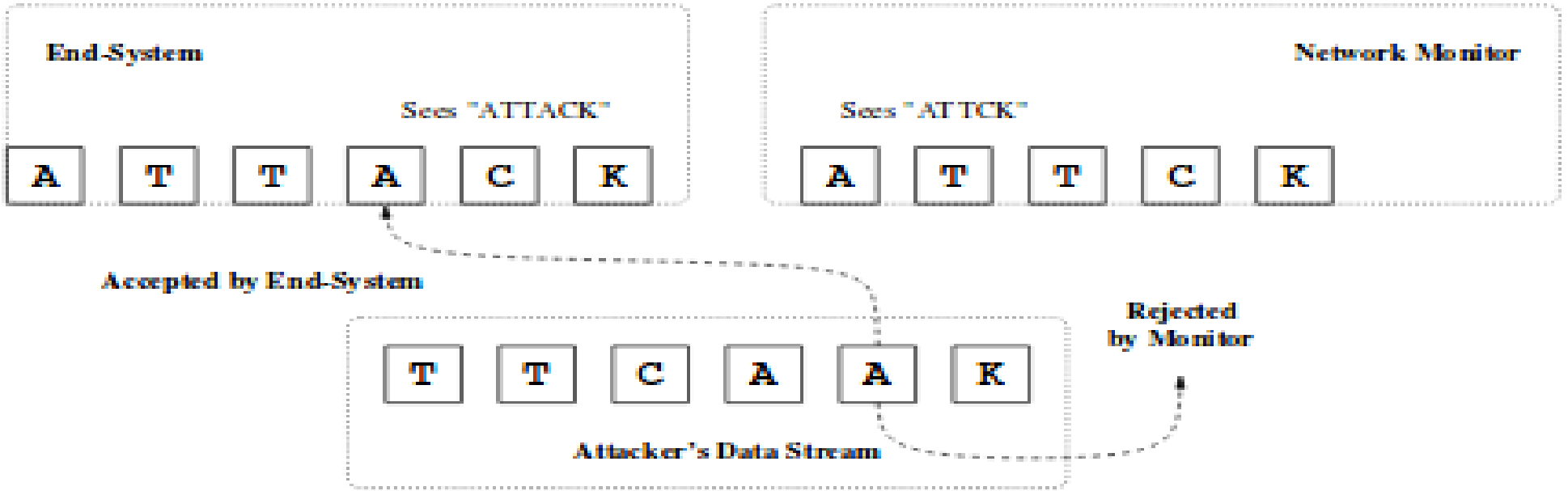


Figure 5: Evasion of the letter 'A'

Figure from Ptacek and Newsham

“Information only has meaning in that it is subject to interpretation”

–Computer Viruses, Theory and Experiments by Fred Cohen, 1984

“The only laws on the Internet are assembly and RFCs”

–Phrack 65 article by julia@winstonsmith.info

“Information is inherently physical”

--(*Lots of people said this, but see Richard Feynman's Lectures on Computation*)

A layer 7 example (XSS) due to Jeff Knockel

- Suppose “`<script>...</script>`” is blacklisted
- Use “`<script>...`” instead, many browsers will happily run the script anyway despite the missing closing tag
- Information only has meaning in that it is subject to interpretation
  - IDS interprets things one way, web browser another

Physical layer injection

- From https://www.usenix.org/legacy/events/woot11/tech/final_files/Goodspeed.pdf

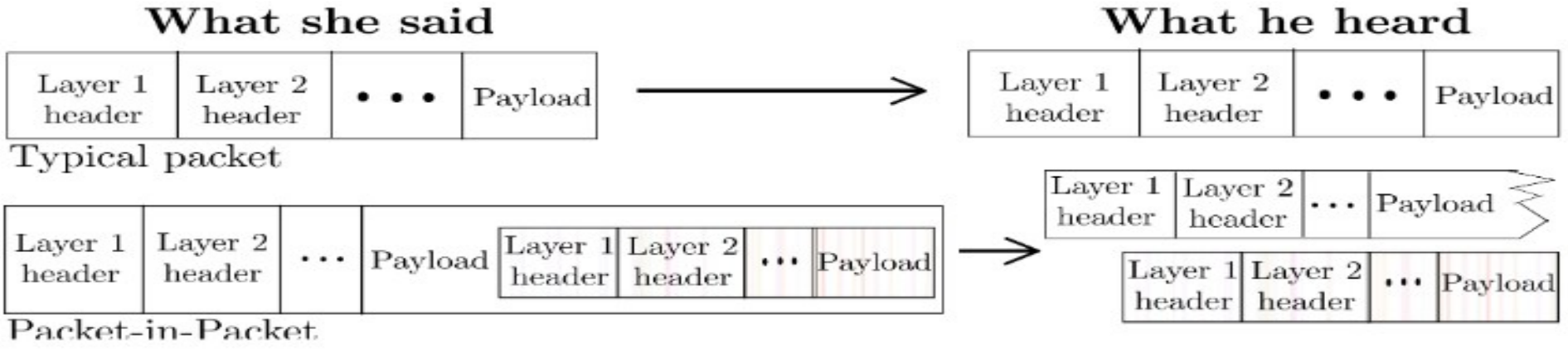
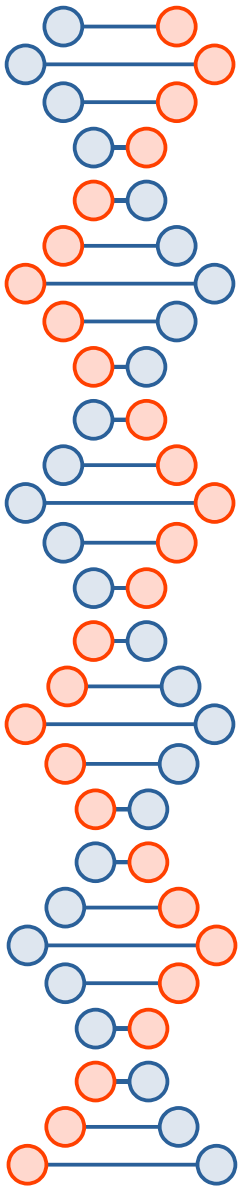
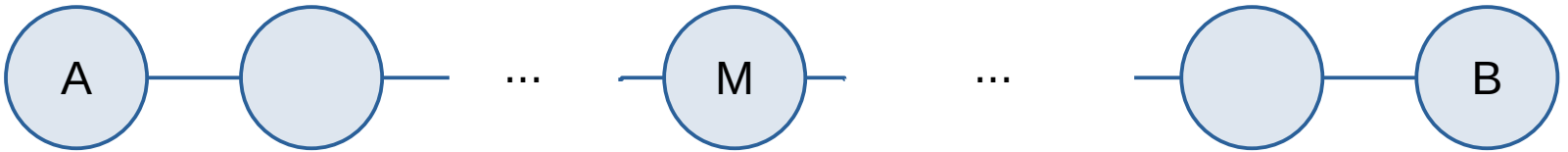


Figure 2: A typical packet's interpretation contrasted with that of a PIP.

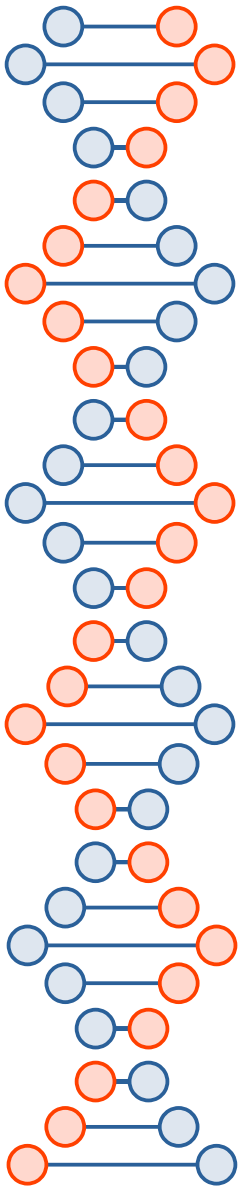
Denial-of-Service (DoS) for IDS

- Exhaust the IDS's resources in some way
  - CPU
  - Memory
  - Bandwidth
- Fail-open (just let stuff through) *vs.* fail-closed (slow down the network)
- Example: On accident, “Tony” brought down the UNM Computer Science Dept. network




A
...
M
...
B

A, B, and M may implement the protocol differently



Long answer: still NO!

- A, B, and M may implement the protocol differently
  - Postel's law: be conservative in what you do, be liberal in what you accept from others
- It gets worse… The Internet is designed end-to-end, there is a lot of state in A and B that M is not privy to
  - "Correctness" is defined as a partial ordering in distributed systems
- It gets worse… Even if they all agreed on the protocol, it's impossible even for A and B to know the state of the connection they share unambiguously