

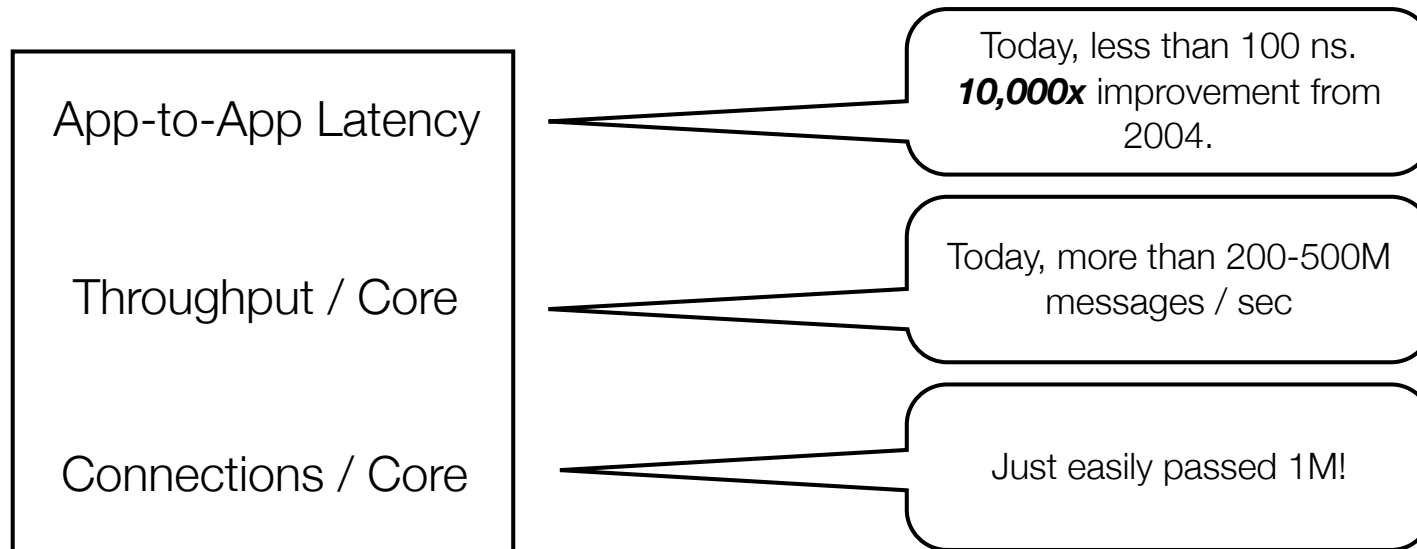
Building High Performance Protocols

Todd L. Montgomery

@toddlmontgomery

Informatica Ultra Messaging Architecture

Protocol Design & Implementation



👉 *Cost*, 👉 *Capacity*
👉 *Profit*

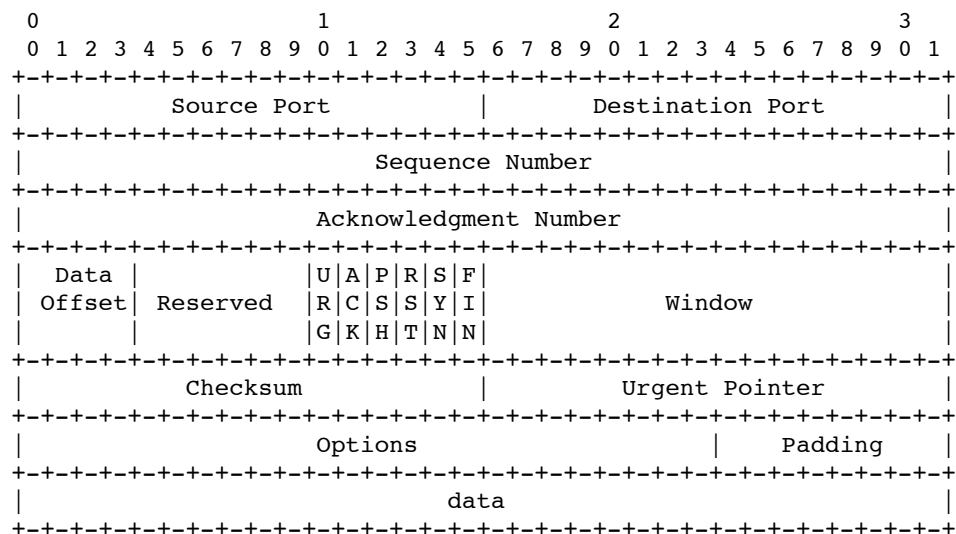
pro·to·col *noun* \ˈprō-tə-,kəl, -kōl, -käl, -kəl\

...

3 b : a set of conventions governing the treatment and especially the formatting of data in an electronic communications system <network *protocols*>

...

3 a : a code prescribing strict adherence to correct etiquette and precedence (as in diplomatic exchange and in the military services) <a breach of *protocol*>



Data Format & Layout

```

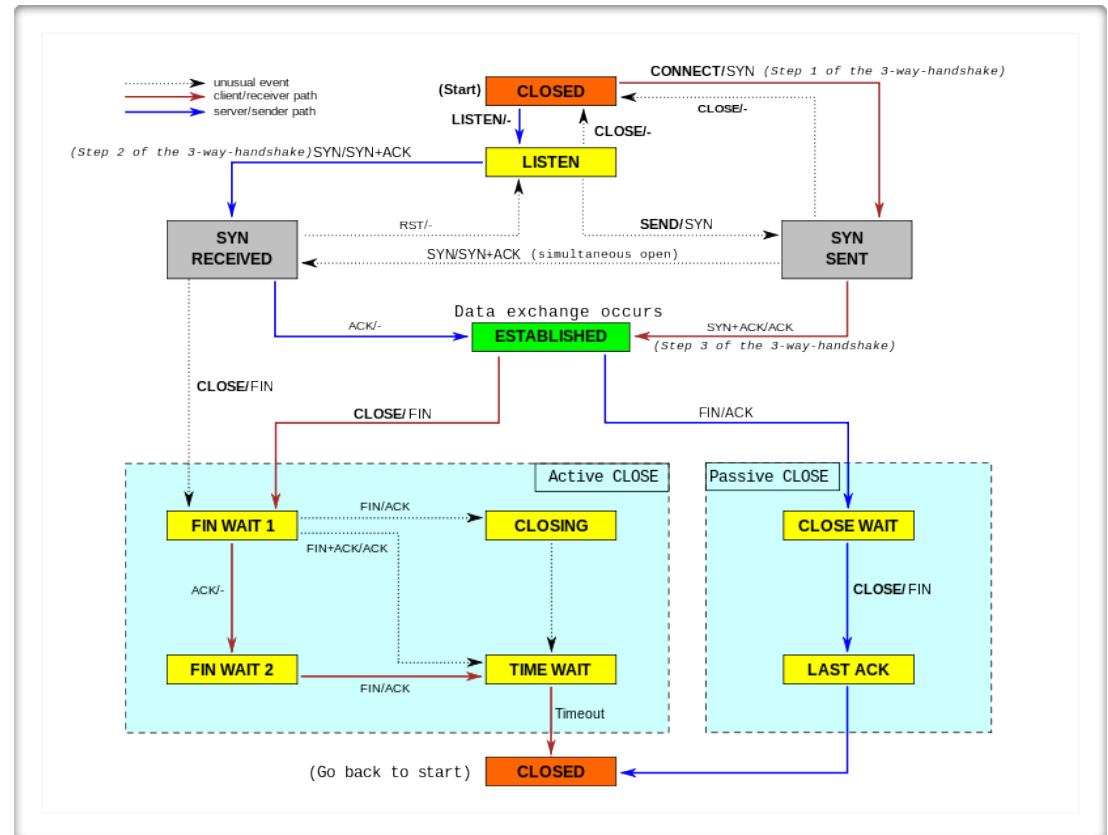
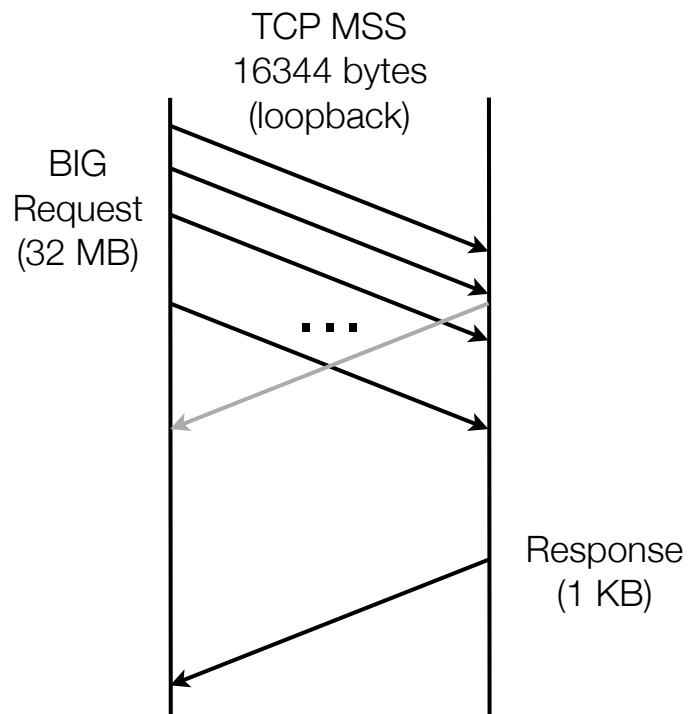
0000030: 6555 5409 0003 291f ad50 a925 ad50 5578 eUT...)..P.%.PUx
0000040: 0400 980a 980a ecfd 093c 94ed f738 8edf .....<...8..
0000050: 6306 631d 8a52 5242 a548 844a 9628 2315 c.c..RRB.H.J.(#.
0000060: 1ac4 b420 b28d c916 33da 281a cab8 d3be ... ....3.(.....
0000070: efd2 be6a 9336 1185 a8b4 8954 a4ed d6a8 ...j.6.....T....
0000080: 14a1 c8fd 3fd7 3da3 7ade cff3 bc97 cff7 ....?.=.z.....

```

```

8=FIX.4.2 | 9=178 | 35=8 | 49=PHLX | 56=PERS | 52=20071123-05:30:00.000 | 11=ATOMNOCCC9990900 | 20=3 | 150=E | 39=E | 55=MSFT | 167=CS
| 54=1 | 38=15 | 40=2 | 44=15 | 58=PHLX EQUITY TESTING | 59=0 | 47=C | 32=0 | 31=0 | 151=15 | 14=0 | 6=0 | 10=128 |

```



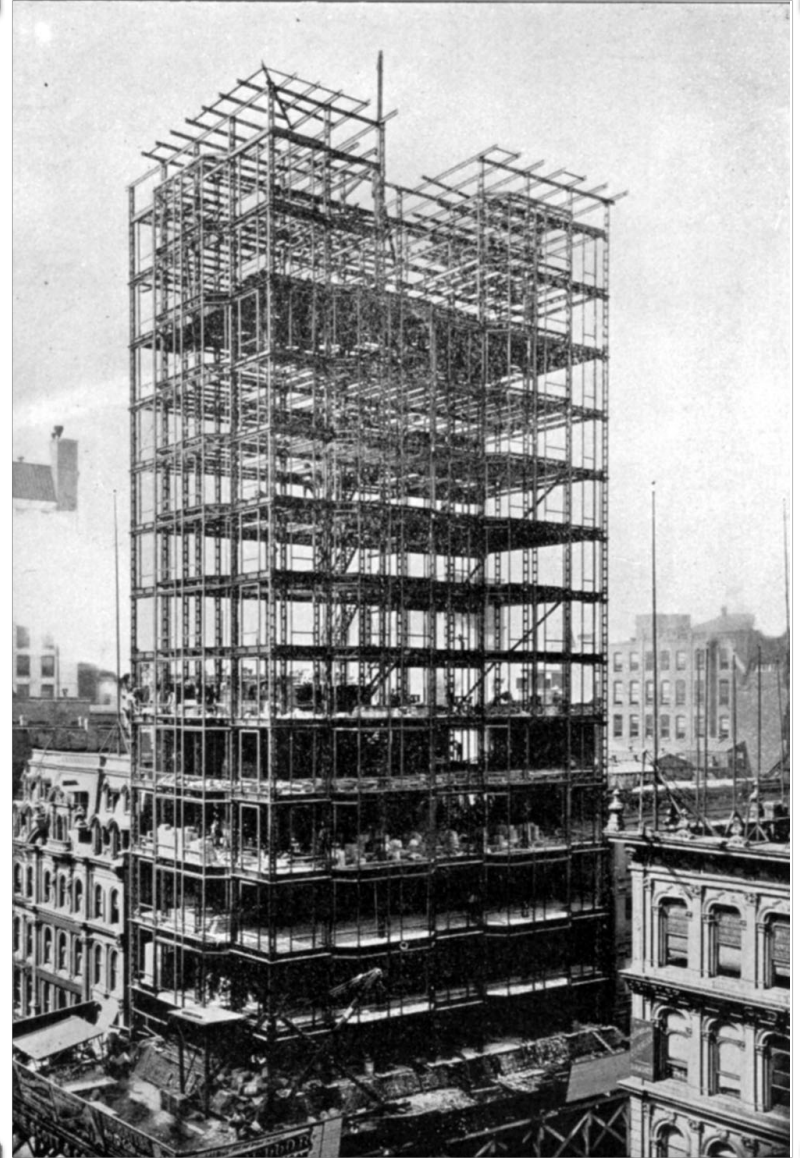
Data Exchange



Compatibility

Backwards

Forwards



Implementation

Intimately Tied!

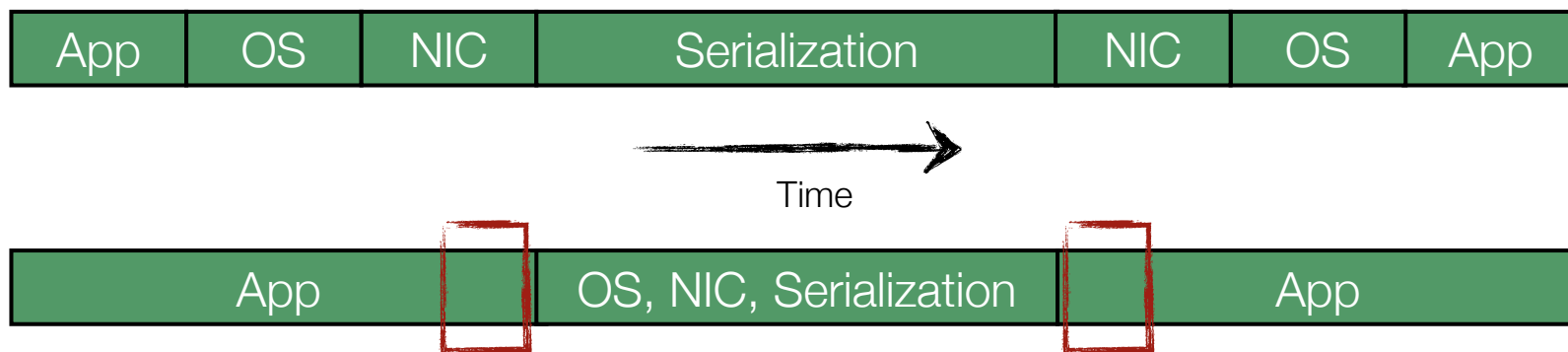
Data Format & Layout

Compatibility

Data Exchange

Implementation

Where does all the time [and CPU] go?



It's an array!



It's All About the Arrays

- ▶ Individual datagram or a stream, don't care
- ▶ Binary or ASCII, don't care
- ▶ Leverage CPU architecture, language, OS, etc.
- ▶ Leverage striding & access patterns
- ▶ Leverage cache lines

***Mechanical
Sympathy!***

Binary vs. ASCII

Binary Layouts

Myths

- ▶ Parsing is hard
- ▶ Fixed size fields always too small
- ▶ ...

Reality

- ▶ Overlays/Casting can make it simple
- ▶ Serializing fields can be simple
- ▶ Byte ordering is straight forward
- ▶ Fixed size fields are a Good Thing
- ▶ Always ways to add more Types
- ▶ Fields are easy to validate



ASCII Layouts

Myths

- ▶ Parsing is easy (lots of libs to help)
- ▶ Parsing is slow
- ▶ Text very easily extended
- ▶ ...

Reality

- ▶ Parsing can be “fast” (x86 SIMD)
- ▶ Often have to touch every byte
- ▶ No static field size “hampers” laziness
- ▶ Much harder (and slower) to validate
- ▶ Extension can be a hairball

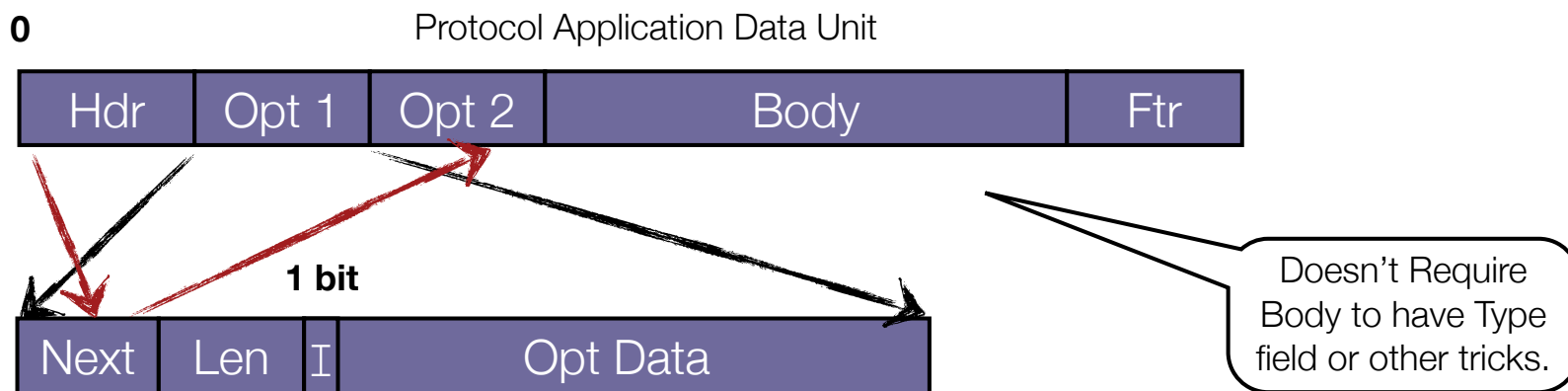
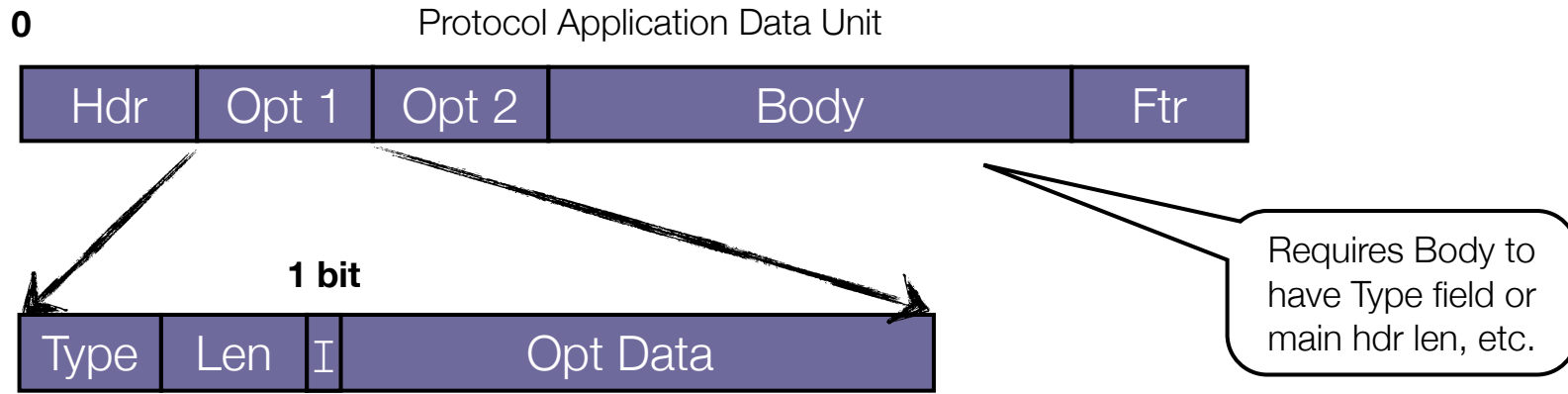
*Always work with the hand you are dealt!
Sometimes you can't change the protocol
Seldom is ASCII or Binary == (black || white)*

Layout & Striding

```
00004f0: 27cf 5c08 726b 8da7 486d f305 8e18 8727 '.\rk..Hm.....'
0000500: 07ba 9b14 18e9 90da ce20 8569 6d49 1b2c .....imI.,
0000510: 0b02 a02b 5095 cb25 5f11 76b8 1ae2 13d4 ...+P..%_.v.....
0000520: 2148 8924 2220 1e30 e325 5f71 44e5 98c4 !H.$" .0.%_qD...
0000530: 621b 0a55 e068 4ad3 01d0 0259 4845 8028 b..U.hJ....YHE.(
0000540: 0999 5cbe e2ac cca4 6a31 bbc2 b2b6 e520 ..\.....j1.....
0000550: ce7e 86fb d4e3 cdf8 f7c2 b76a 14ad 62ff .~.....j..b.
0000560: aec2 776a f4cf f46f 99ee cfc4 6a8b 7682 ..wj...o....j.v.
0000570: 6270 af16 1576 8bbe 39b1 56c9 81f1 218d bp...v..9.V....!.
0000580: 3277 1b3b 62de 1ca2 37b4 d218 a706 51f2 2w.;b...7.....Q.
0000590: a680 bd8d 7f05 2b35 1882 dea4 7607 d0d1 .....+5....v...
00005a0: c885 770e 91d3 4d92 ae90 bb18 9e8d 15bd ..w...M.....
00005b0: 3154 b266 1c94 bc80 de89 1f50 a5a8 83b6 1T.f.....P....
00005c0: 9c0e 3dc6 21b5 d391 f2d9 0929 a4b0 82d4 ..=.!.....)....
```

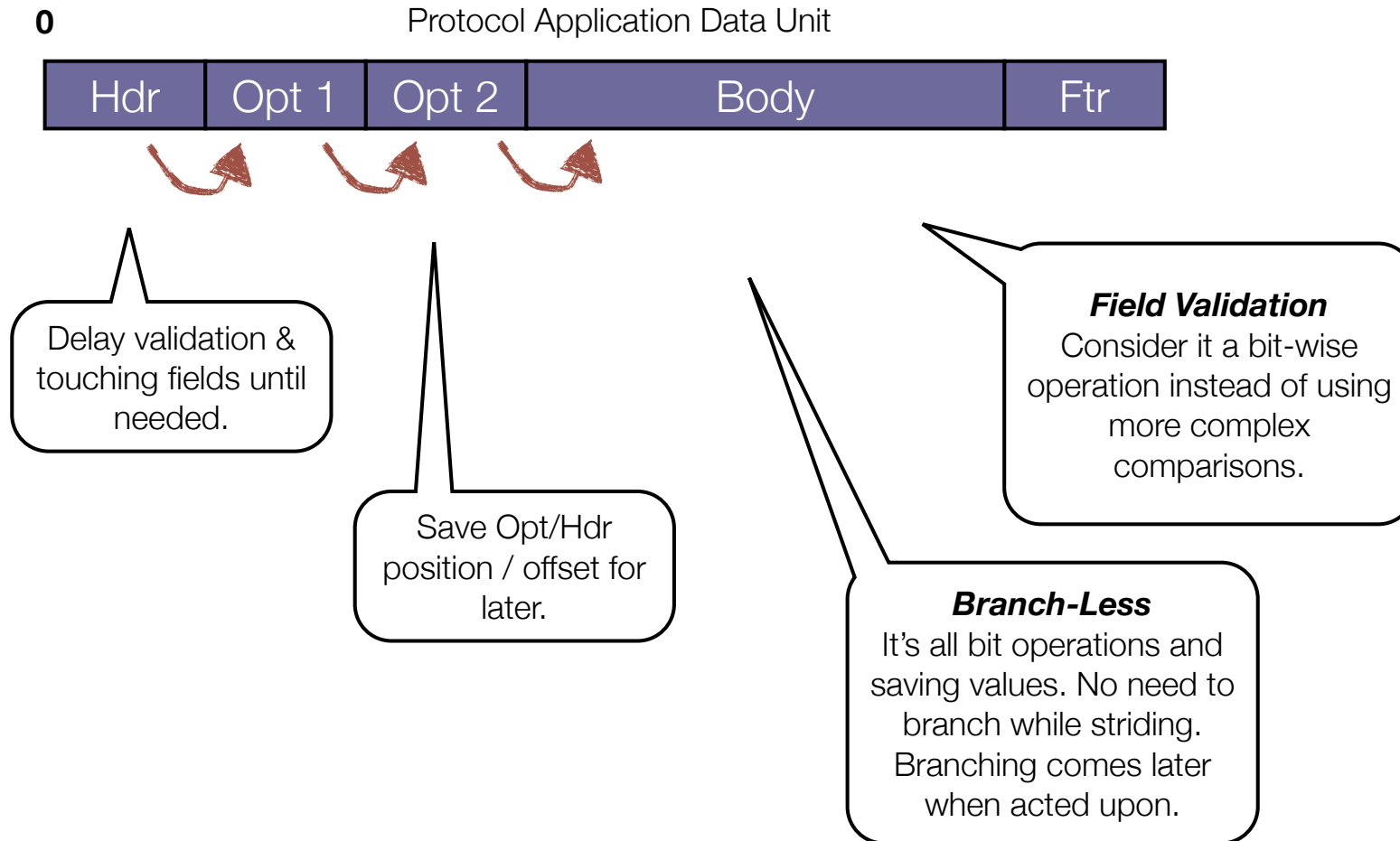
- ▶ Access patterns for fields are important! Design them in!
- ▶ Which fields are touched in which order for common case?
- ▶ How do cache lines (64-byte) align with access pattern?
- ▶ Will this layout allow for *predictive* striding line-to-line?
- ▶ However... What if you are stuck with a layout?

Header Chaining



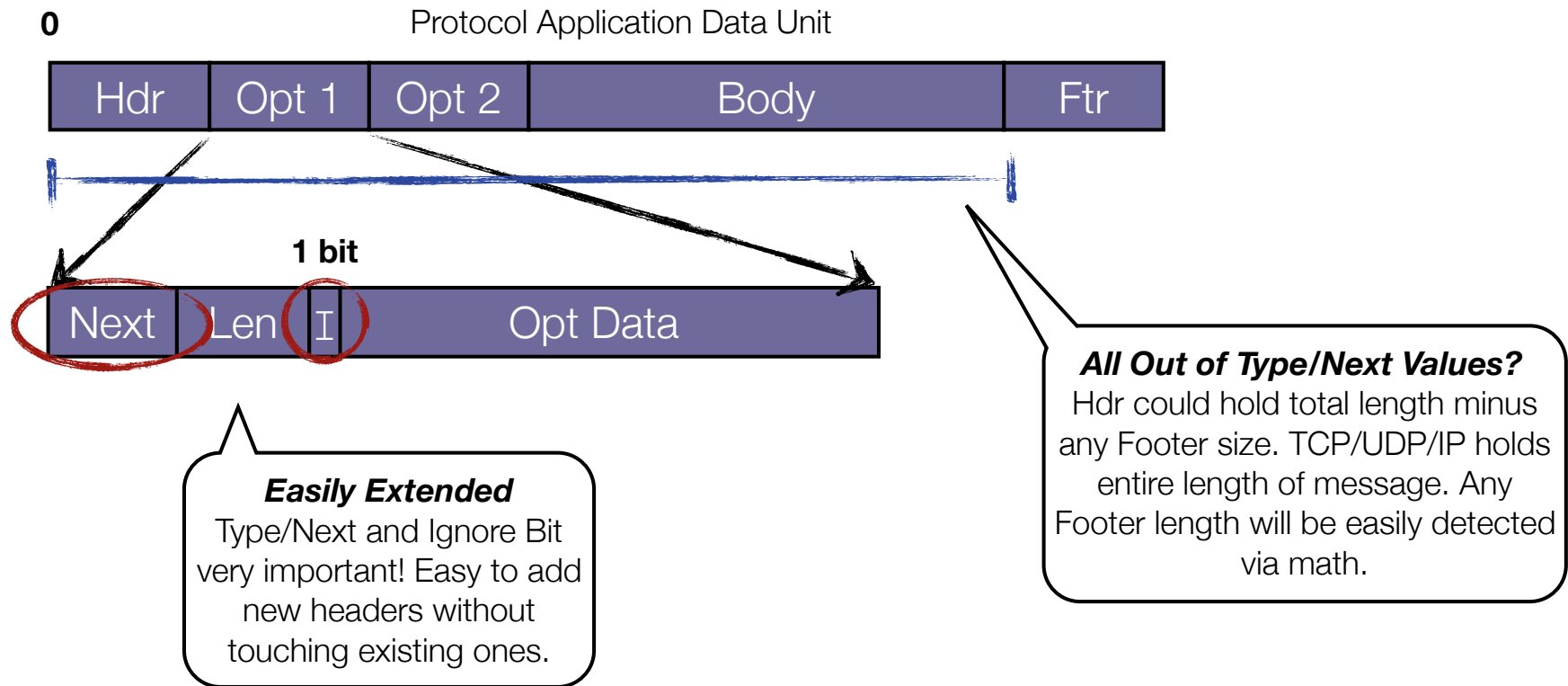
Looks like it is designed for striding?

Lazy Header Striding



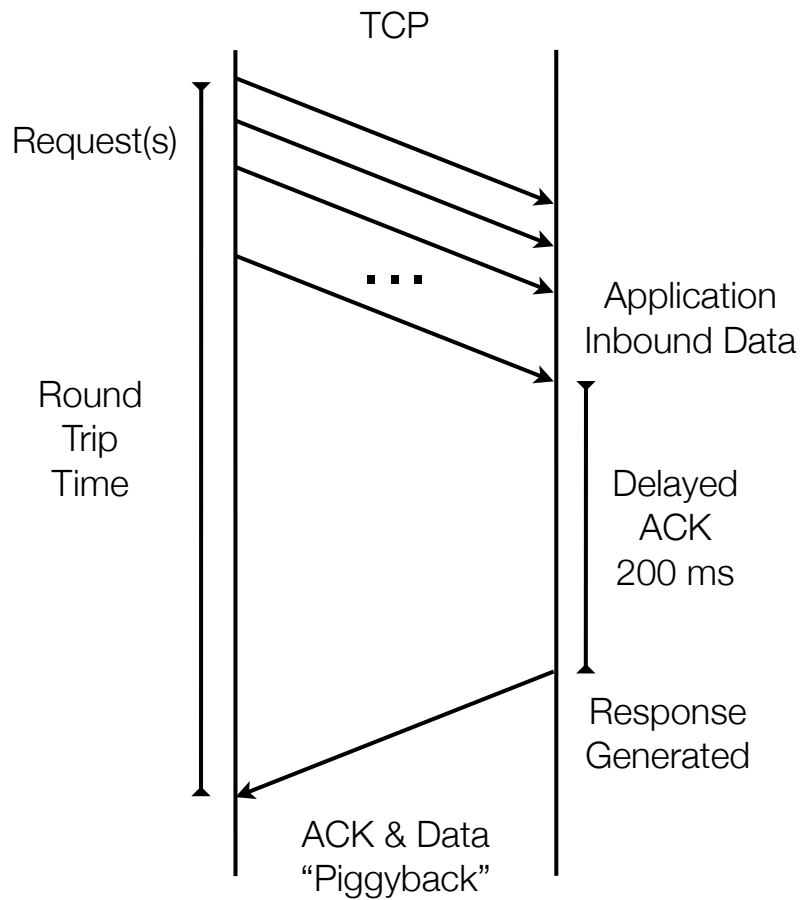
Some fields or entire options/headers may not be needed for processing... yet

Compatibility



Extending binary formats is always possible with some handy tricks

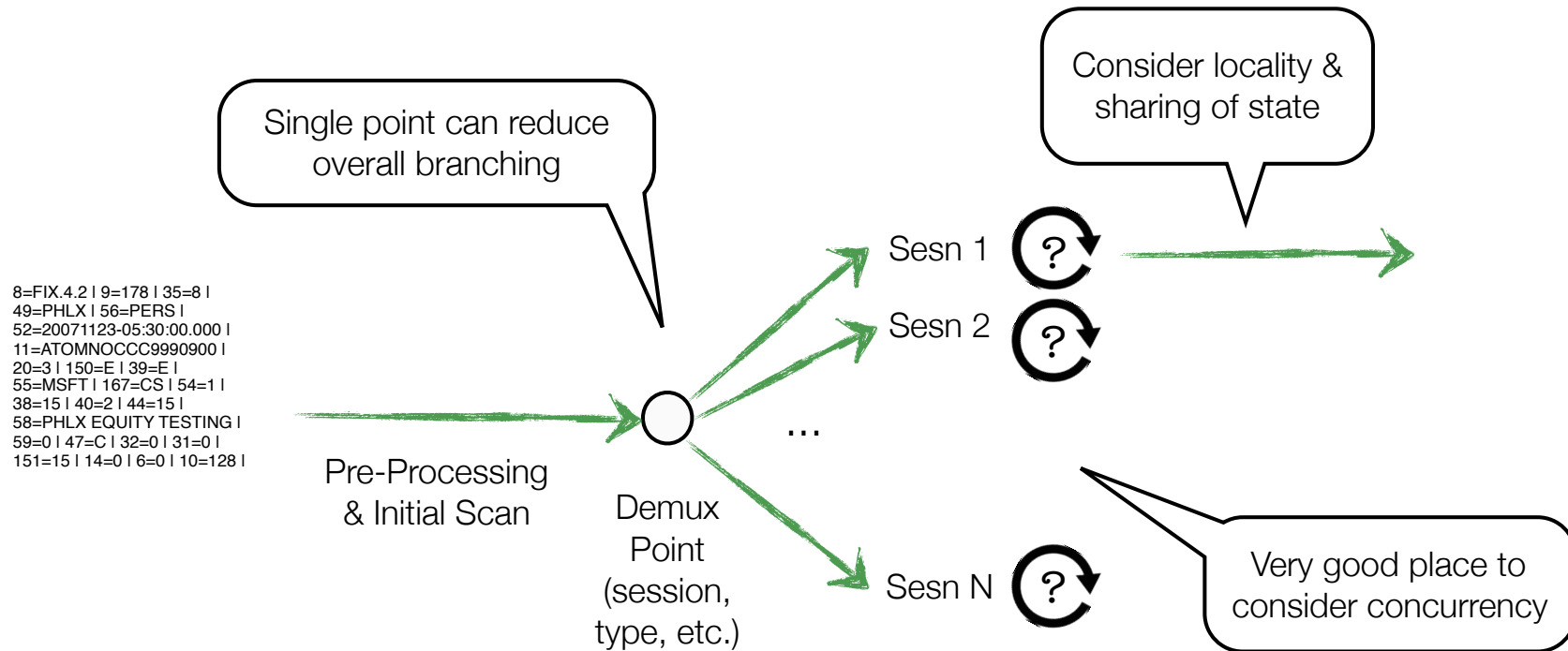
Request/Response



Leverage Piggybacking

- ▶ TCP provides 200 ms to respond
- ▶ One less message in an exchange
- ▶ Applies for responses-to-responses

Plumbing



Contention

- ▶ Consider arriving data to be immutable
- ▶ Copy on read?
- ▶ Copy on retain? (stack-based)

HTTP to SPDY

Modifying HTTP

Reduce web page load time

- ▶ Multiplexed Requests
- ▶ Prioritized Requests
- ▶ Compressed Headers
- ▶ Server Pushed Streams

Control order of page load and optimize display using only a single connection

In addition, avoid sending duplicate headers unless they have changed

Proactively send oft-requested content

HTTP 2.0?

SPDY is the foundation!

IPv4 to IPv6

Simplify Router Processing

- ▶ Simpler basic header
- ▶ No fragmentation
- ▶ No header checksum
- ▶ Options extensibility (Next Header chain)
- ▶ Rename TTL → Hop Limit

No Fragmentation

- ▶ Permanent Don't Fragment (DF)
- ▶ Endpoints do Path MTU Discovery
- ▶ Default min MTU of 1280 octets

No Header Checksum

- ▶ Link & Higher layer integrity protection
- ▶ UDP required to have own checksum

Routers do less work per packet, easier to implement ASICs, higher switching speeds!

Less is more!

Questions?