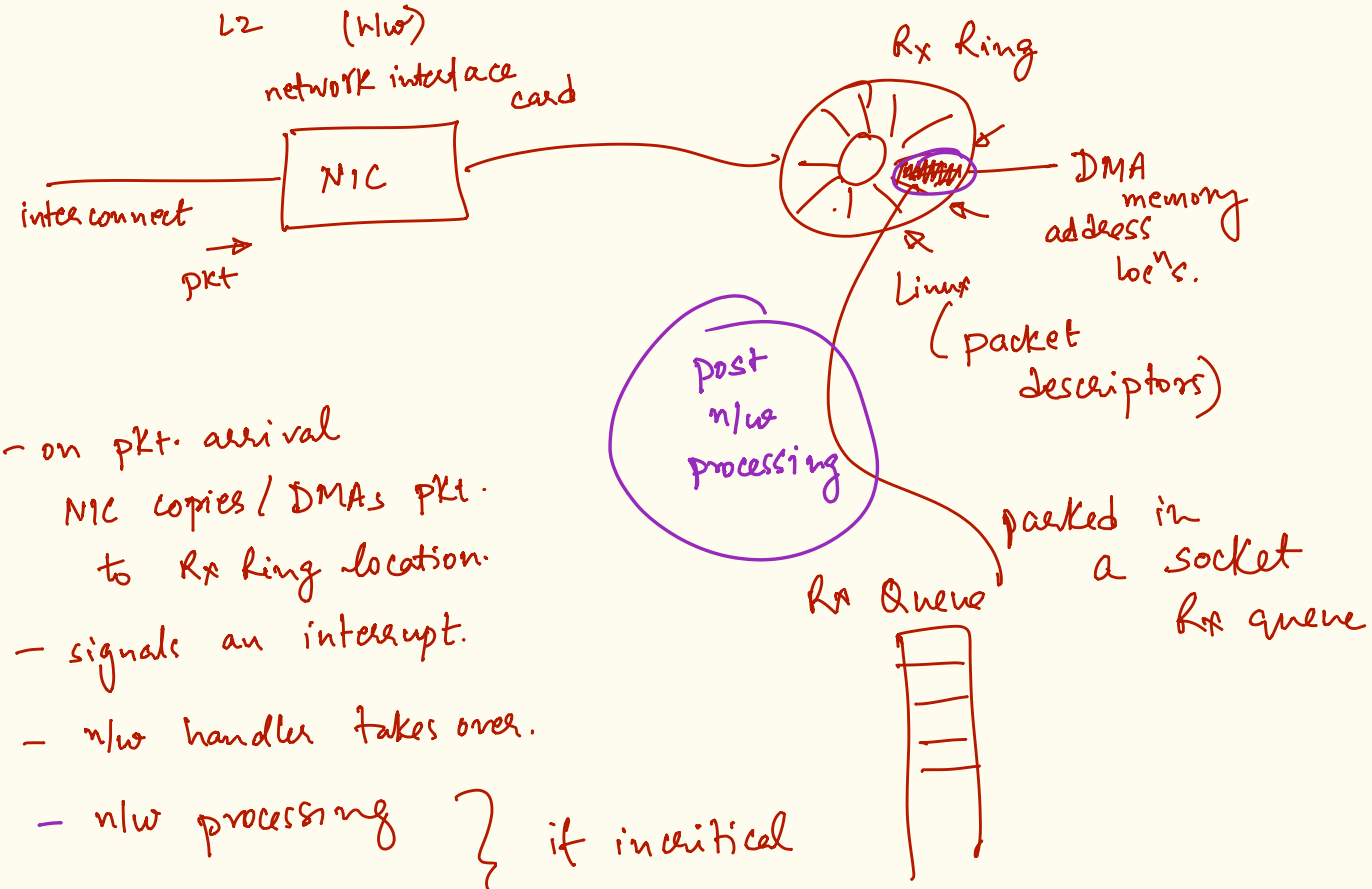
CS 744 Lecture 22

network IO

- interrupt-driven and/or polling-driven.
- top half vs. bottom half.



- on pkt. arrival
  - NIC copies / DMAs pkt. to Rx Ring location.
- signals an interrupt.
- n/w handler takes over.
- n/w processing + socket mapping } if in critical path can take quite a lot of time.

design principle: split processing.

step1: handle interrupt, send ack to NIC, schedule fn thread to handle rest. enable interrupts. (top half).

bottom half. step2: bottom half – packet processing.

- protocol header processing
- apply iptable rules
- handle TCP state & actions
- reassemble pkts.
- deliver to socket!

softirq

Ksoftirqd

schedulable Kernel thread (high priority)

adaptively use interrupts vs polling.

low rate

high rate.

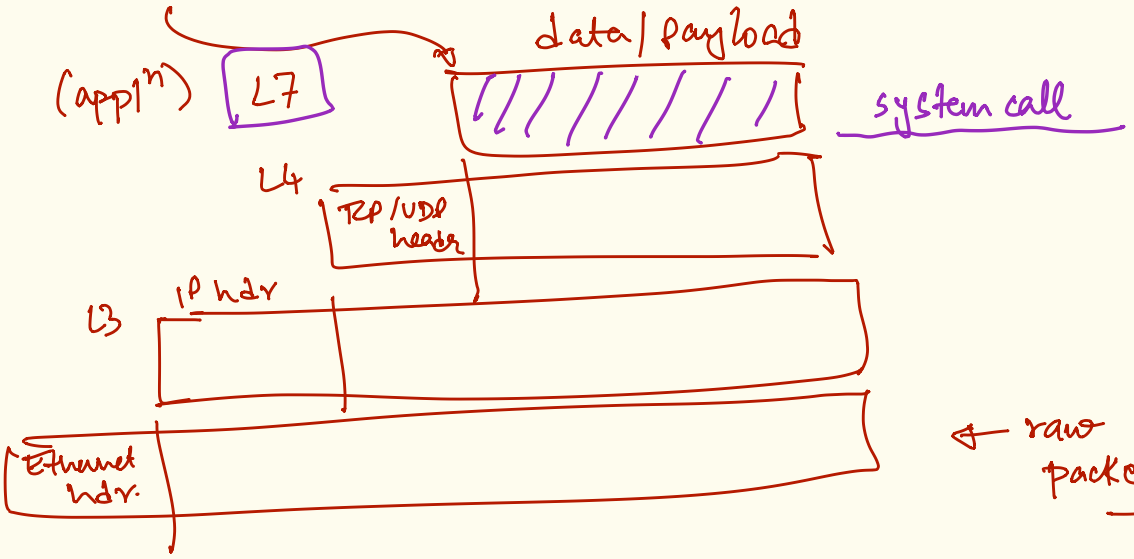
Linux NAPI

new API

SK_buff. ~ socket buffer

OS structure to handle/wrap packet metadata & contents.

```
read(sockfd, buf, len);
```



web server example : HTTP protocol | L7 protocol | application-layer

request

type - GET, POST, HEAD

args - name of resource (html file)

response || data/payload in n/w pkts.

msg. response

header + contents.

response type

length

:

(*) http : stateless protocol

- each msg/req. is a new/independent msg.
- no implication on action due to previous requests.

as compared to being stateful

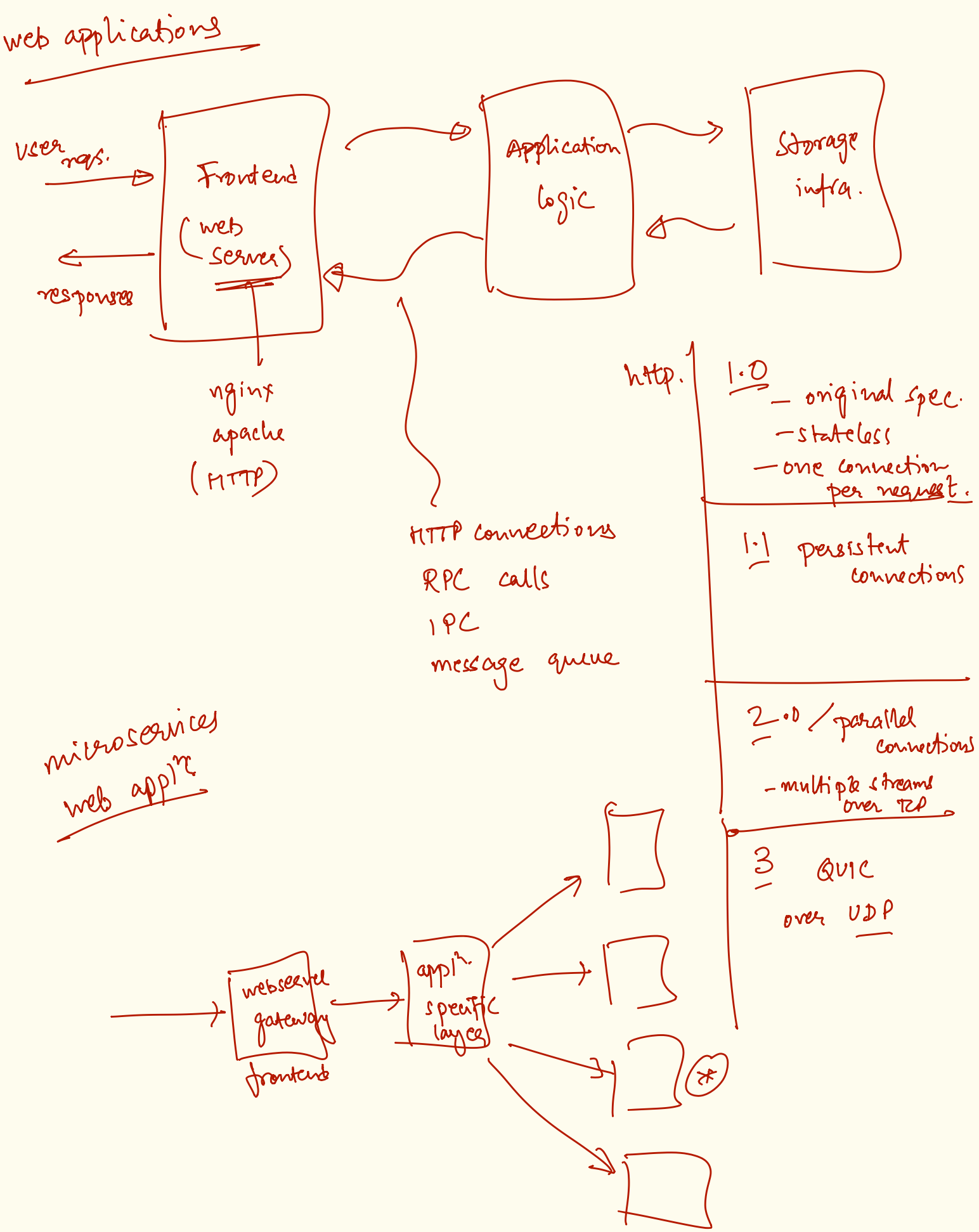
e.g: TCP

whose protocol actions are dependent on prev. actions msgs.

- reliability
- retransmissions
- RTT est.

current day statefulness (is) appln driven wrappers.

- session
  - session id is negotiated as part of appln.
  - all http msgs. embed session-id in request } which is interpreted by business logic.
- cookies
  - leaving behind bread crumbs!
  - cookie info. attached to http requests.


web applications
user reqs.
Frontend
(web servers)
responses
Application logic
Storage infra.
nginx
apache
(HTTP)
HTTP connections
RPC calls
IPC
message queue
http.
1.0
- original spec.
- stateless
- one connection per request.
1.1 persistent connections
2.0 / parallel connections
- multiple streams over TCP
3 QUIC
over UDP
microservices web applns
webserver gateway
frontend
appln. specific layer

RPC ~ remote procedure calls.

- "remotes" a local fn. call.

```
{
    resize(image);
}
```

- hijacks the fn. call
- lifts it to execute remotely (on a n/w ed machine)
- & returns result

RPCs ~ transport of fn. info.

- fn. type/name
- fn. arguments — int, pointers, structs } serialization, deserialization

gRPC

RPC stubs

serialization
de-serialization.

specification formats

- JSON
- XML
- ProtoBuf

} mechanisms to encode arguments/info.