Communication:

Sender

send (s/w)

→ h/w →

Receiver

receive (s/w).

Active Messages:

theory: Log P.

L → latency

O → overhead from s/w

g → max msg rate (h/w capability)

P → processors

RTT = 4O + 2L.

How to lower overhead:

→ remove OS from data path ⇒ user level networking.

Adv

performance.

↳ mainly due to trap, mem mgmt (copy).

Disadv

virtualization

protection (someone else receiving ur msg) (mux/demux).

U-net:

1) low overhead.
2) protection.
3) unreliable.
4) not req/response.

Why user-level ?

→ performance
→ flexibility.
→ don't have to be general.

Architecture of U-Net:

⇒ Endpoint

→ Communication segment has msg data.

→ Queues (used by client and NI).

i) Send : put the msg in comm segment; put msg descriptor in send queue.

2) Recv

3) Free : used by NI to grab a free communication segment.

⇒ Protection.

→ Tag : to identify the endpoint.

→ use OS to tag

→ during communication, no OS.

→ On Receive end U-Net performs demux

which process recvs msg?

Base U-Net:

→ limited size comm segment. pinned in memory

↳ no swapping.

→ not true zero copy : 1 copy.

→ small msg optimization : put msg in queue ; not involving comm segment.

Kernel emulation:

→ one OS managed endpoint; OS virtualizes endpoint.

⇒ prob: again involves OS and traps.

~~Later solution~~ → VM integration with endpoints.

Hardware: -100 -200.

↓

programmable NI.

→ host has mem, NI has mem ⇒ Which mem to put queue in?

→ send queue in NI

→ free queue in NI

→ recv queue in host.

Note the stair steppy graph in figure 3:

→ Each ATM pkt is 48 bytes (steady for that).