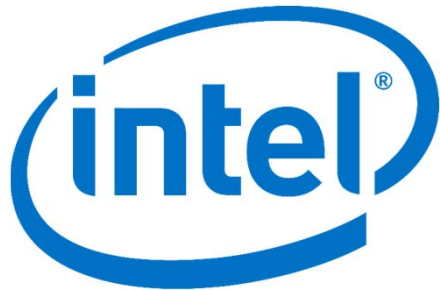


presented by



Improving UEFI Network Stack Performance

Spring 2019 UEFI Plugfest

April 8-12, 2019

Presented by Maciej Rabeda (Intel) and Vincent Zimmer (Intel)

www.uefi.org

Agenda



- Motivation
- Starting point
- Design limitations
- What can be done?
- Tackling the problem
- Results





Motivation





Motivation

- Legacy BIOS is still there
 - Cost vs benefits of transitioning to UEFI from [Legacy](#)
- UEFI network performance complaints
- [Linuxboot](#)

```
SeaBIOS (version rel-1.9.1-0-gb3ef39f-prebuilt.qemu-project.org)

Initializing Intel(R) Boot Agent XE (X550) v2.4.31
PXE 2.1 Build 092 (WfM 2.0)

Intel(R) Boot Agent XE (X550) v2.4.31
Copyright (C) 1997-2018, Intel Corporation

CLIENT MAC ADDR: D0 0D E0 00 00 02  GUID: 00000000 0000 0000 0000 000000000000
DHCP._
```

Real world feedback to address this issue

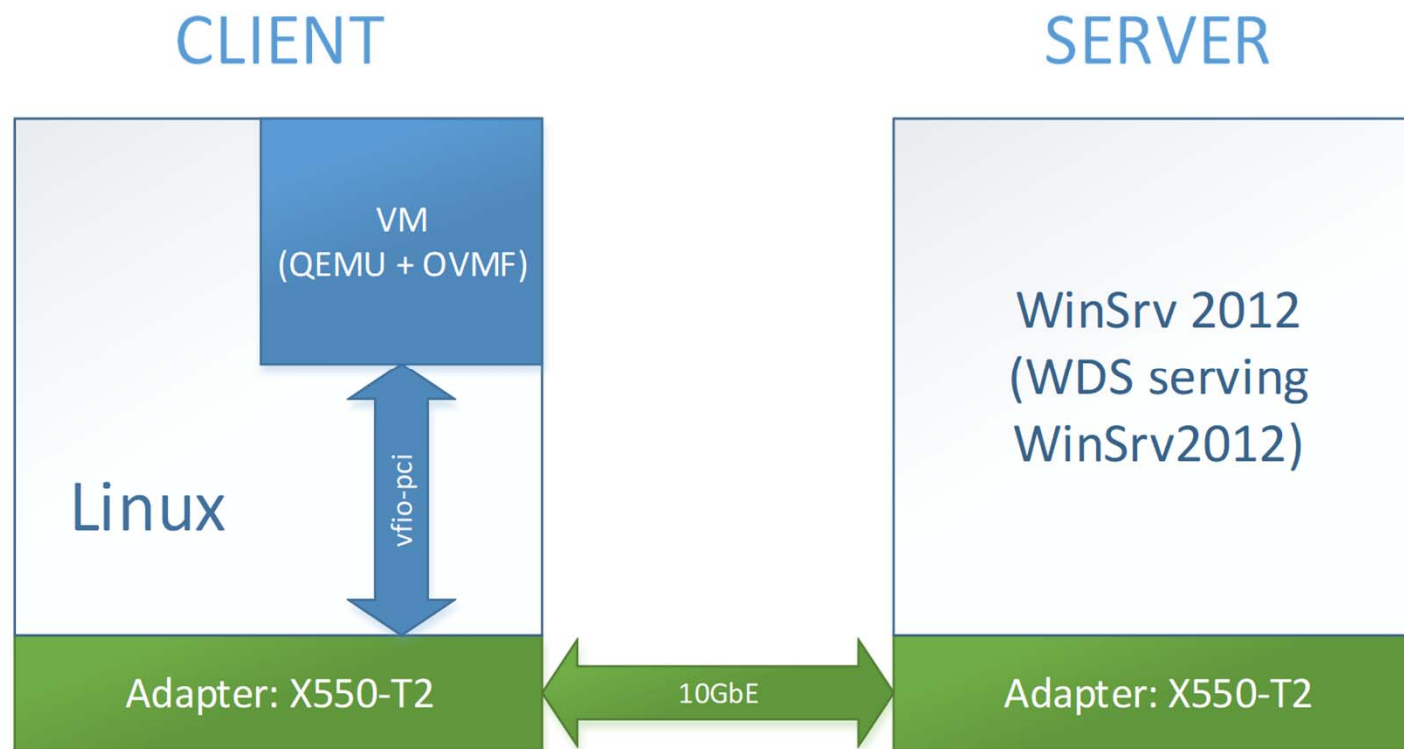
Starting point

- How good/bad is it really?
- PXE boot to [WDS](#)
 - Legacy vs. UEFI
 - Adapter: Intel(R) X550-T2
 - Client directly connected to WDS
- Measure boot.wim transfer time with Wireshark



Photo by [Tim Gouw](#) on [Unsplash](#)

Test setup



Starting point - numbers



Technology	Platform	Environment	Avg. time[s]	Avg. speed[Mbps]
Intel(R) Legacy PXE 10G	Bare metal	UEFI CSM	2.1	~790
	QEMU	SeaBIOS	5.5	~300
UEFI PXE	Bare metal	UEFI	9	~185
	QEMU	UEFI (OVMF 2018 DEBUG)	8.4	~200
	QEMU	UEFI (OVMF 2018 RELEASE)	3	~550

Optimized UEFI is still behind

Identifying the sky



- Test: UEFI networking limits
 - Server: Tx packet flood to client, full MTU, fixed number of packets (10 million)
 - Snp->Receive() loop: 10 Gbps
 - Mnp->Receive() loop
 - ARP & IP unloaded 4.5 Gbps
 - IP unloaded 4 Gbps
 - PXE BC DL 0.55 Gbps



Observations

- OEMs building stack drivers in DEBUG mode (instead of RELEASE)
- Extra receivers on MNP layer reduce Rx throughput
 - Packet processing through whole stack before receiving another packet
 - Event and DPC cost
- Improvement potential with current stack driver:
 - 4 Gbps on MNP but only 0.55 Gbps on PXE BC



What can be done?

- Culprit
 - It's probably our stack (same WDS serving legacy and UEFI install images)
- Optimization opportunity?
 - Is it stack complexity alone?
 - Still does not address whole packet processing before receiving next one



The problem areas identified



But I want to play...

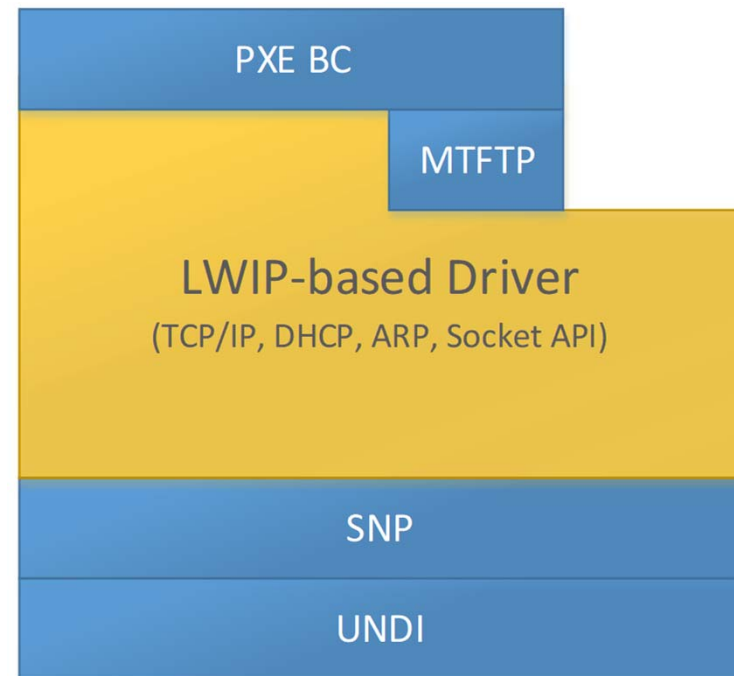
Play time

- Idea: multiprocessing in UEFI networking
- Feasibility:
 - Got EFI_MP_SERVICES_PROTOCOL
- Benefits:
 - Network offload (partial or full) from Boot strap processor (BSP) to Application Processors (Aps)
 - Performance scalability
- Concerns:
 - Thread safety (core, stack, protocols...)
 - Current network stack complexity
 - UEFI spec conformance

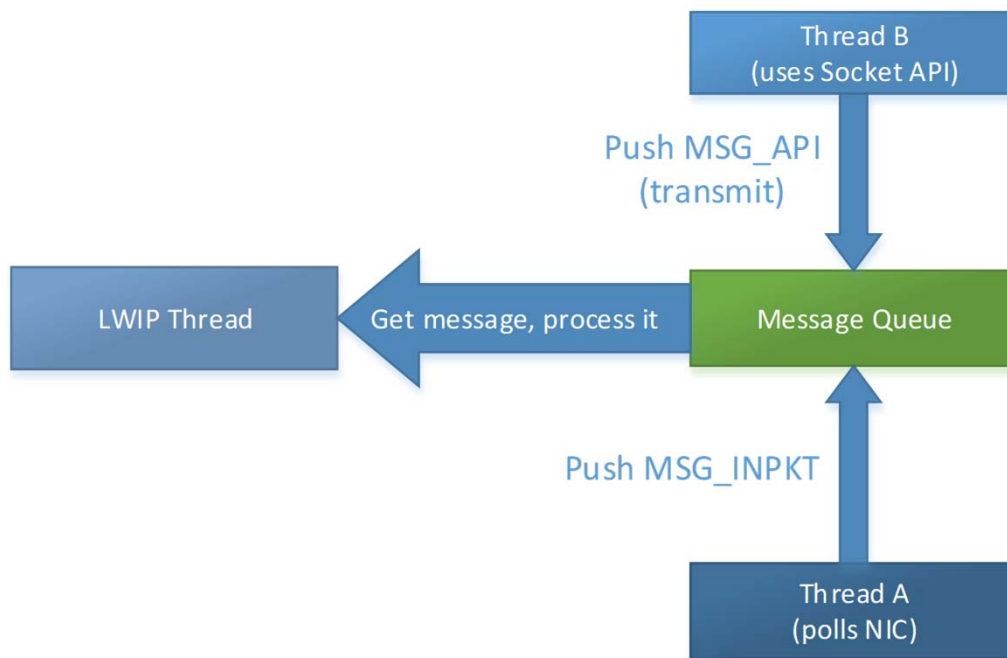


Design assumptions

- Provide an alternate TCP/IP stack to an existing one
- Offload whole networking from BSP to APs up to OS-like socket layer
- Use an existing open-source, thread-safe TCP/IP implementation
- Use SNP for interfacing to network adapters
- Expose DHCP API (needed by PXE BC)

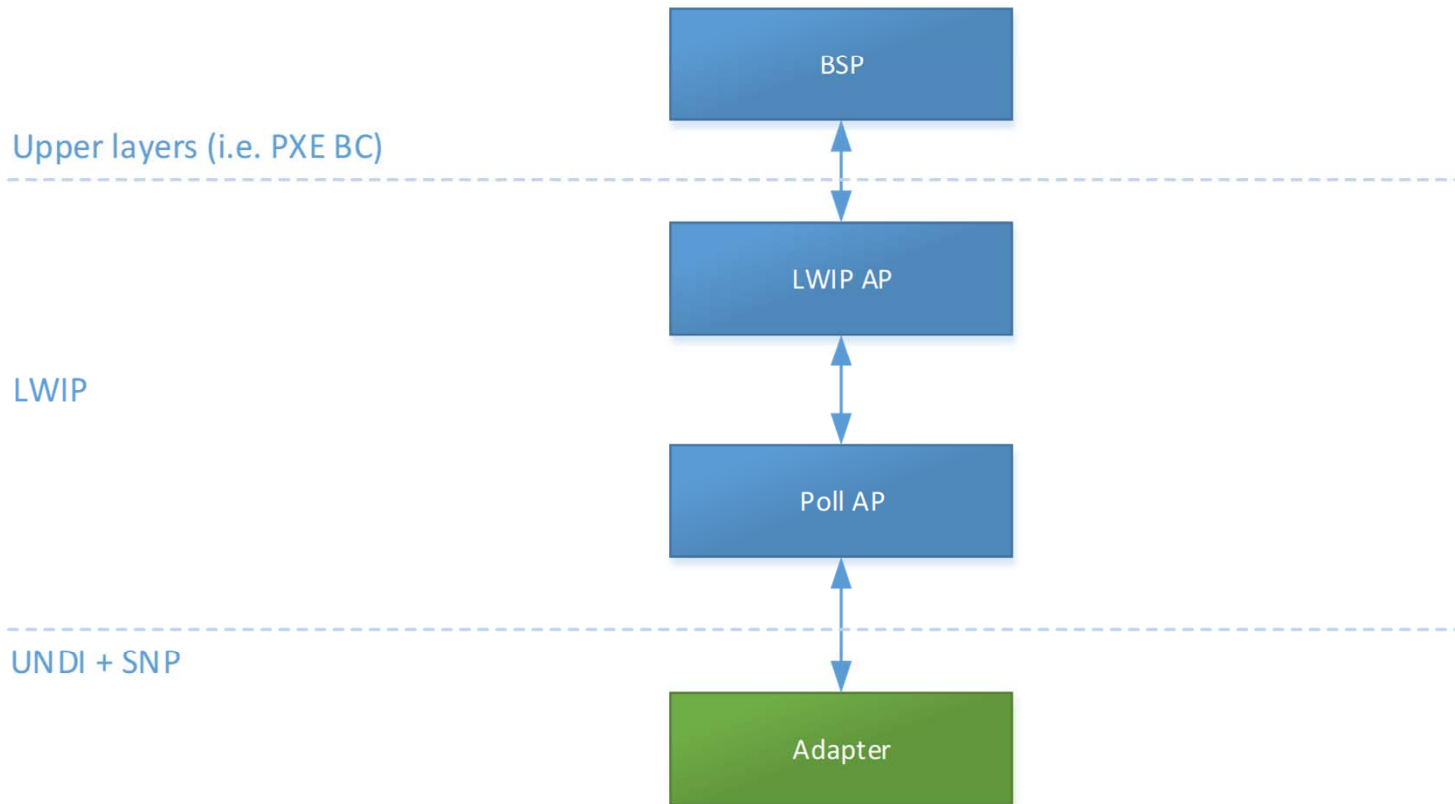


TCP/IP stack

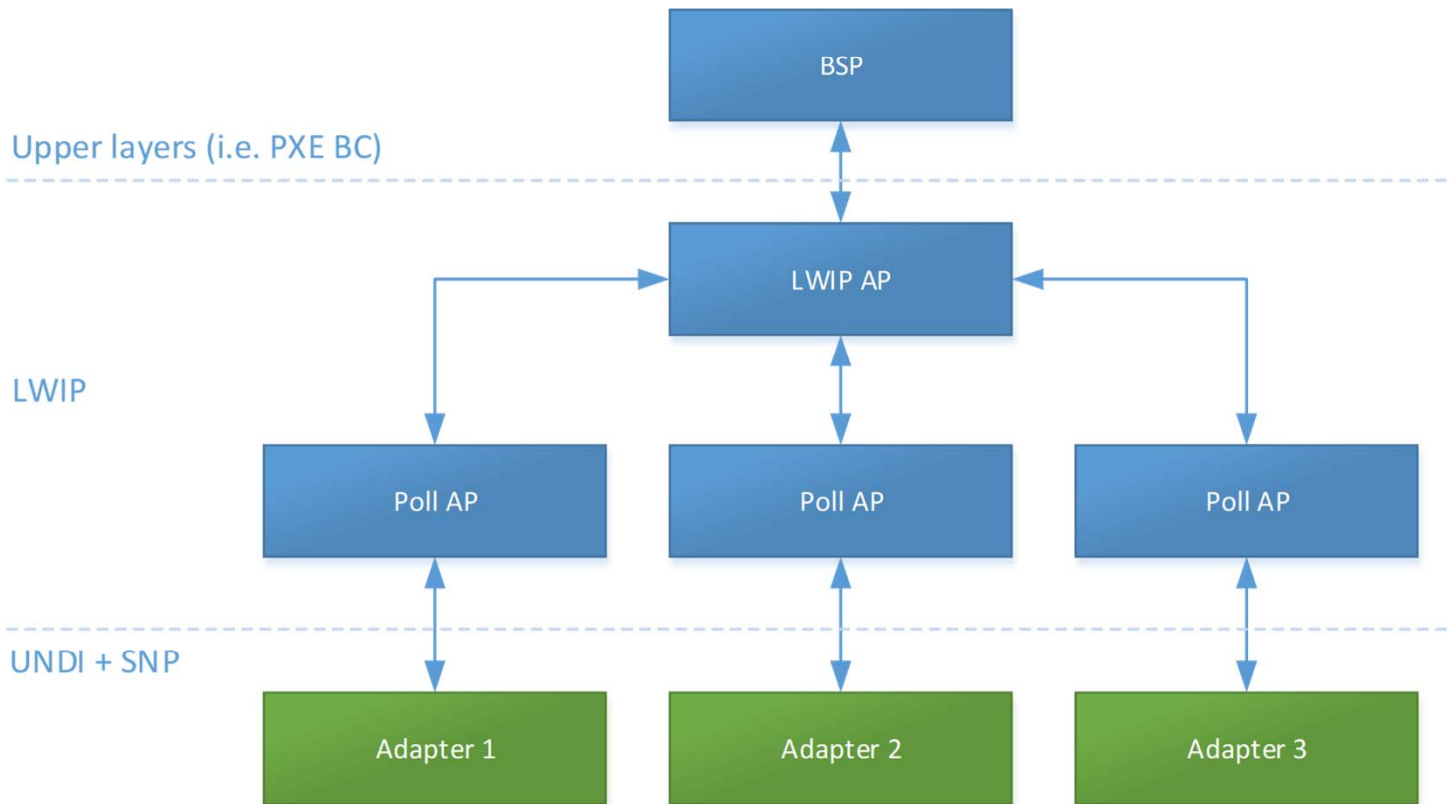


- Implementation used: lwIP
 - Uses single thread for all packet processing
 - Thread-safe at API levels thanks to message mechanisms
 - Exposes OS-like socket layer
 - Almost autonomous - needs minimal changes to make it work with UEFI

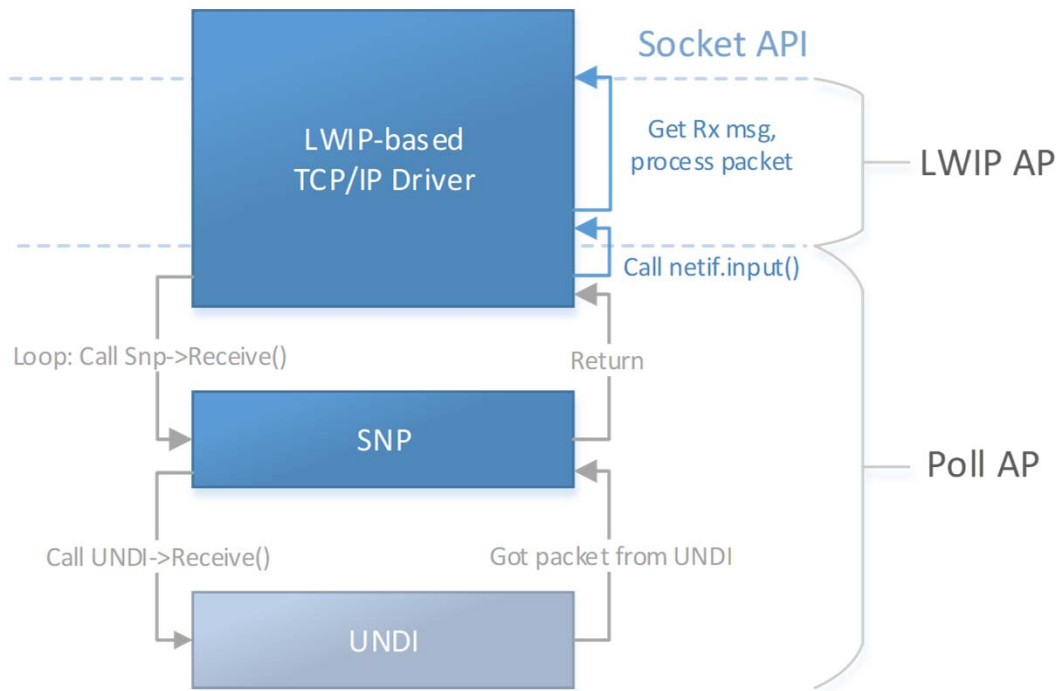
Multiprocessing



Multiprocessing

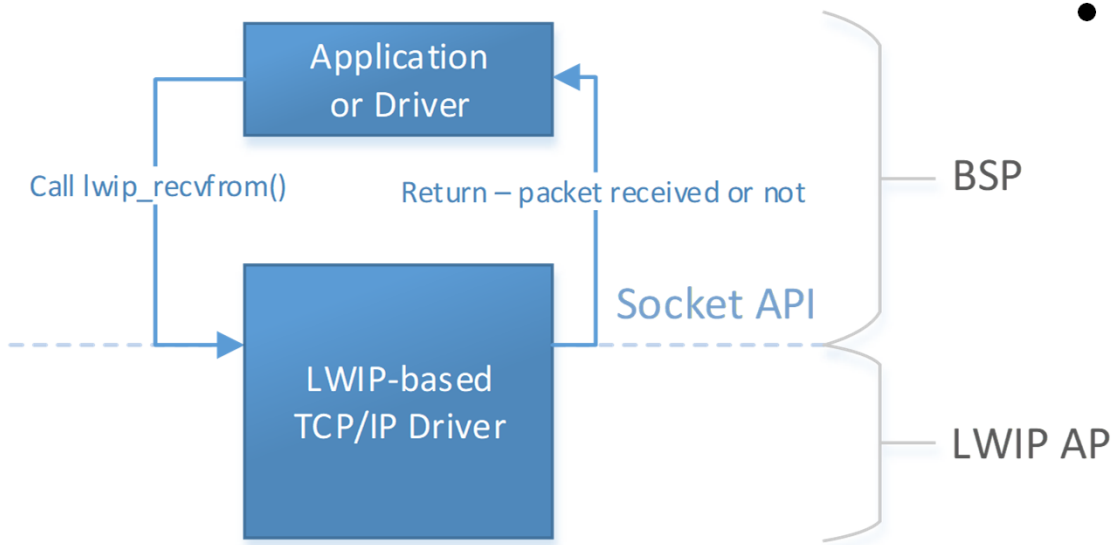


Receive path – APs



- Poll APs:
 - Interface SNP in busy-poll
 - Once packet is received, send msg to LWIP AP to process
- LWIP AP:
 - Process the packet
 - Assign to proper sockets

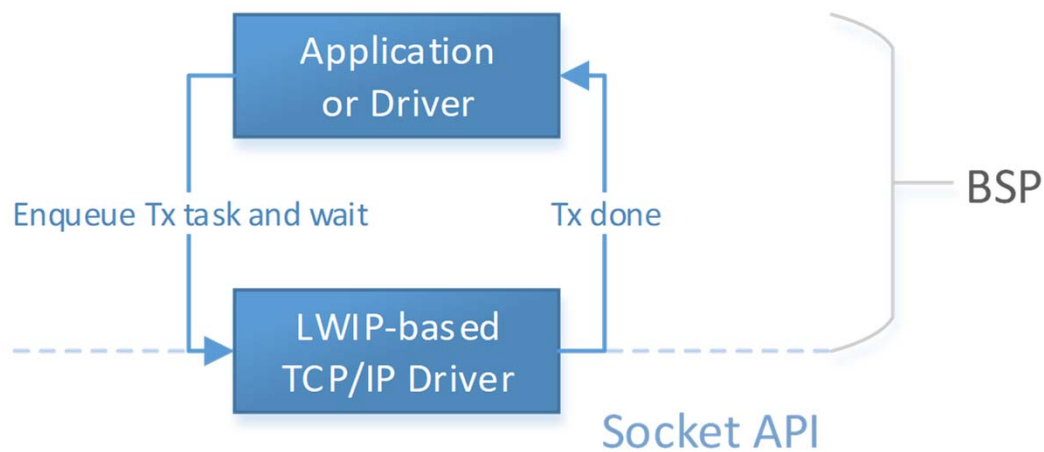
Receive path – BSP



- BSP
 - Loop `lwip_recvfrom()` to get packets from socket

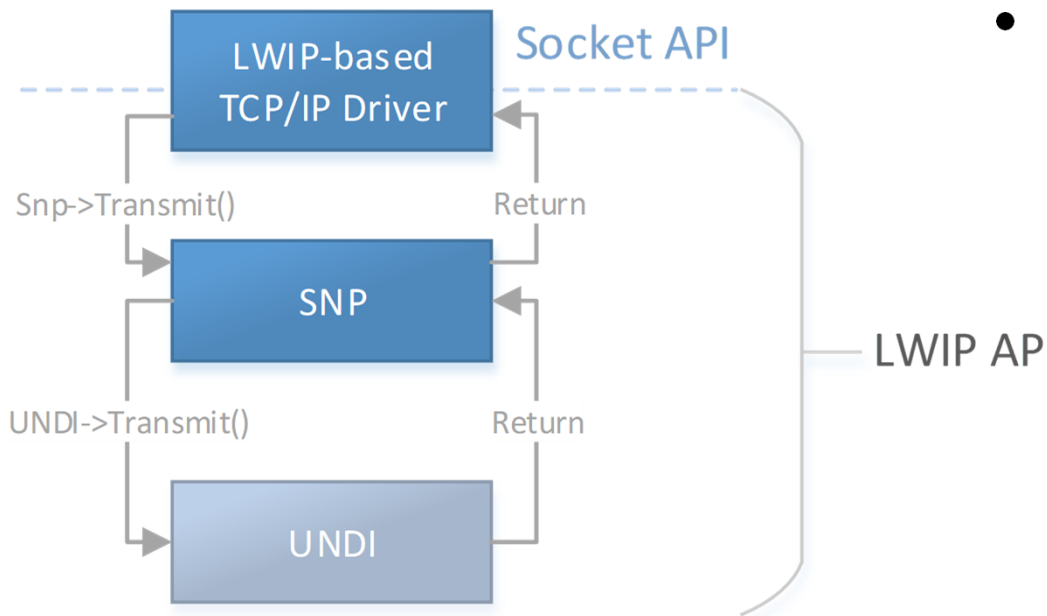


Transmit path – BSP



- BSP
 - Send message to LWIP AP to send a packet
 - Wait for LWIP AP to finish the transmit

Transmit path – AP



- LWIP AP

- Triggered with a message
- Call SNP->Transmit() directly
- Signal the waiting application Tx has finished

Summary of changes



- New

- MdeModulePkg/Universal/
 - Network/MpTcpIpDxe
 - ThreadingDxe
- MdeModulePkg/Library/
 - DxeThreadingLib
 - DxeThreadingStructLib

- Modified

- MdeModulePkg/Universal/
 - Network/Snp4Dxe
 - Network/Mtftp4Dxe
 - Network/PxeBc4Dxe
- OvmfPkg/Library/
 - PlatformDebugLibIoPort

Final tests

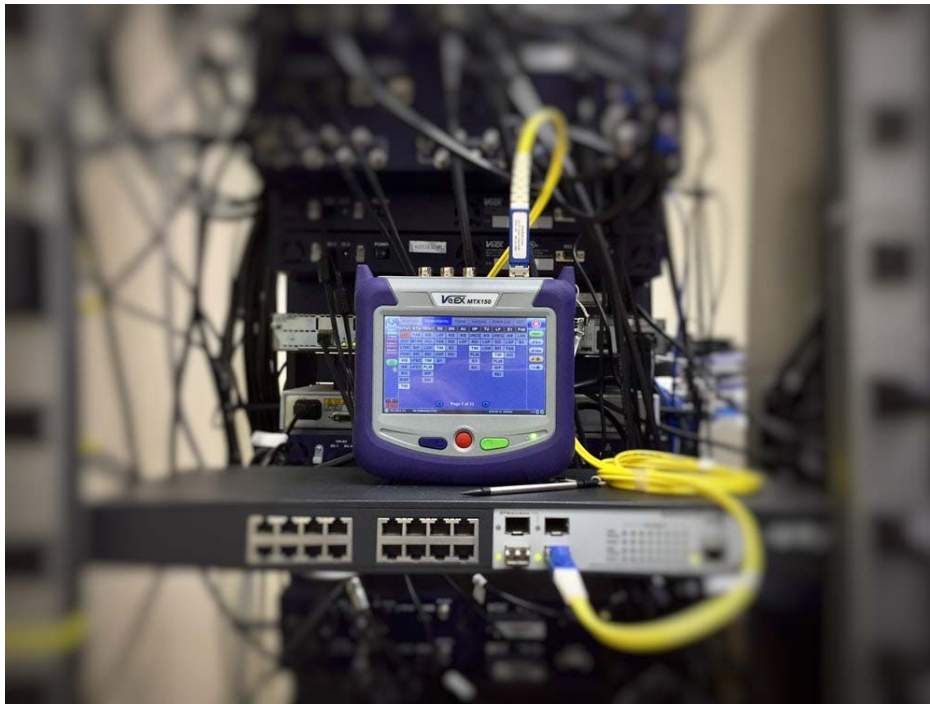


Photo by [Ildfonso Polo](#) on [Unsplash](#)

- After all the changes, I could successfully boot to WDS
- How good it is?
 - Same PXE install image DL time test

Results



Technology	Platform	Environment	Avg. time[s]	Avg. speed[Mbps]
Intel(R) Legacy PXE 10G	Bare metal	UEFI CSM	2.1	~790
	QEMU	SeaBIOS	5.5	~300
UEFI PXE	Bare metal	UEFI	9	~185
	QEMU	UEFI (OVMF 2018 DEBUG)	8.4	~200
	QEMU	UEFI (OVMF 2018 RELEASE)	3	~550
	QEMU	UEFI (OVMF 2018 modified)	1.5	~1100

Modern system software techniques can yield 2X UEFI, 1.5X legacy



Summary

- Proven doable
 - Faster than legacy and current solution
- Alternative rather than substitute
- Perf could be better (~1.1Gbps, ~90kpps)
 - Linux OS does 350kpps on a single core
 - Receive Side Scaling, interrupts, etc.

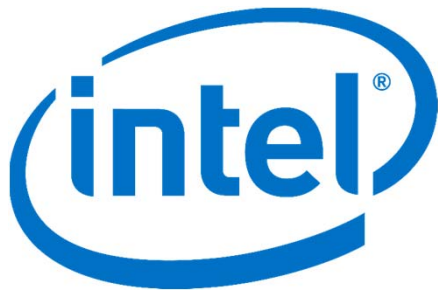
- **Code can be found at:**

<https://github.com/tianocore/edk2-staging/tree/MpNetworkStack>

Thanks for attending the 2019 Spring UEFI
Plugfest

For more information on UEFI Forum and UEFI
Specifications, visit <http://www.uefi.org>

presented by



www.uefi.org

