SILVERENGINE

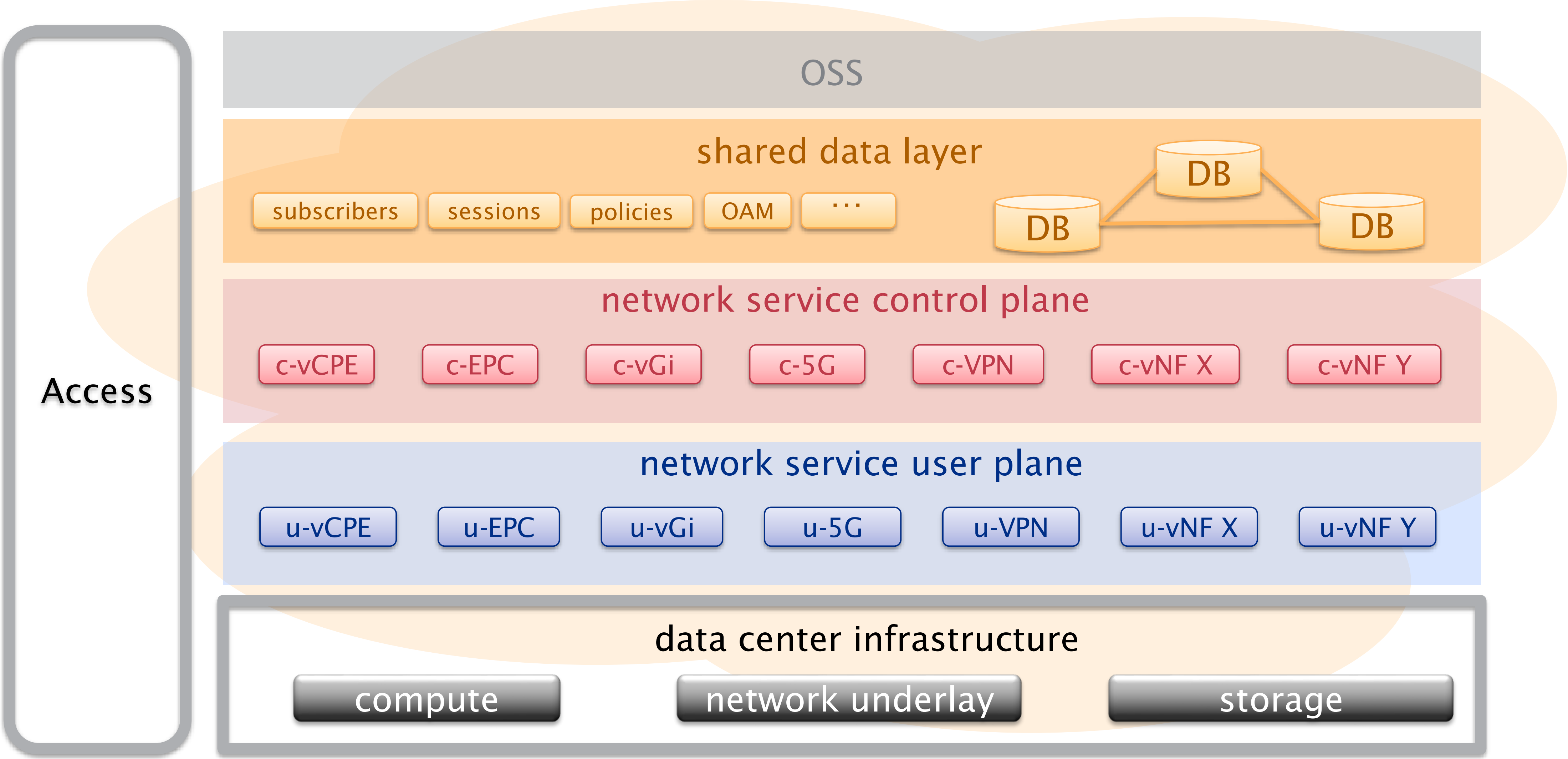
Consultancy, R&D Services and Products for
Mission Critical Information and Communication Technology

network service engine

open source approach to telco networking

May, 2017

vNetwork Structure



traditional service networking architecture

embedded edge products fuse access with service functions into a single box

mobile GW	BRAS/BNG	CMTS	Video edge	MEF switch	VPN PE router	
Internet service	Internet service	Internet service	IPTV service	MEF service	IP/MPLS VPN	
Cellular access	DSL access	HFC access	DSL access	Metro Ethernet	attachment circuits	…

- advantage: access and service specific system optimization
- disadvantages:
    - many vendor specific boxes with different OA&M and life cycles
    - rigid feature set, difficult and expensive to change
    - enormous complexity: ~ O(#accesses * #services)

major disruptions on the way

hitting the traditional service edge vendors

- disaggregation of network elements by fine granular network function virtualization enables radically new service architecture
  - 1:1 virtualization of existing network elements is a dead end
  - makes traditional service edge products (physical or virtualized) obsolete
- 'open source' and IT'zation of networking disrupts the existing business models and value chains
- current IT cloud needs significant networking enhancements to become a Telco cloud
  - X millions of virtual connections/networks, much higher transaction rates, more complex network services

what has changed in networking?

value-add moves from ASICs to open source software running on GPP

- general purpose processors (GPP) can now process *service edge traffic* at negligible HW cost
- no need for ASICs with rigid functions and closed SDKs in a service data plane (but ASICs still essential in packet optical transport underlay)
- data plane*) completely programmable with higher programming languages
- powerful open source available for data plane: Linux kernel
- convergence of compute, storage <u>and</u> service networking on an IT driven Telco cloud platform

*)we use the terms 'data plane' and 'user plane' interchangeably

major user plane options

1	traditional, embedded network elements • ASIC based forwarding plane • proprietary i/f and SDK • closed source SW	• fits for underlay • in DC under pressure by open source white box projects (e.g. Open Compute Project)
2	traditional, cloudified network elements • emulating ASIC of embedded product on x86 • keep proprietary i/f and SDK • re-use of the closed source SW from embedded product	• almost as inflexible as the embedded product • no or limited disaggregation and scale-out
3	'classical SDN' with Openflow or P4 i/f • ASIC based forwarding plane • standardized i/f for a match/action pipeline	• so far not a big success *) • needs new networking software ('re-invent the wheel')
4	open source, native x86/GPP based user plane	• natural fit for cloud environment
4a	• Linux kernel based • cooperative offload options to NICs, FPGA, ASIC co-processors (XDP, switchdev)	• re-use x million lines of code • full disaggregation and scale out support
4b	• Kernel bypass, e.g. Intel DPDK, netmap	• needs new data plane software • higher forwarding performance • re-use of kernel based control plane

*) see https://www.opennetworking.org/images/stories/downloads/sdn-resources/special-reports/Special-Report-OpenFlow-and-SDN-State-of-the-Union-B.pdf

why now ?

IT clouds drive light-weight virtualization technology and new distributed system solutions

- Linux namespace functionality since 2013 (kernel 3.8) rather complete (https://lwn.net/Articles/531114/)
- namespace technology got a major push with “Docker”
- recent advances in cooperative kernel offload (e.g. XDP, switchdev), as opposed to bypass (e.g. DPDK)
- extended Berkeley Packet Filter (eBPF) since E2014 (kernel 3.18) allows for in-kernel flow processing in C with a control plane running in user space
- latest IT advances, like reactive programming (e.g. Akka) simplify concurrent programming task in scale out systems

what others are doing

focus is on the control and management plane for an OpenFlow or P4 controlled data plane

- AT&T is a strong promoter of open source initiatives, e.g.
  - CORD® (Central Office Re-architected as a Datacenter)
  - ONOS®
  - recently open sourcing the ECOMP orchestrator
- however, most of the controllers and orchestrators target an OpenFlow or P4 controlled data plane based on ASICs with match-action-tables
  - requires enormous effort to re-implement the existing IP/Ethernet feature set
  - the missing business case limits so far OpenFlow/P4 adoption in the market*)

*) https://www.opennetworking.org/images/stories/downloads/sdn-resources/special-reports/Special-Report-OpenFlow-and-SDN-State-of-the-Union-B.pdf

OpenFlow (match + action pipeline) is a misfit for a GPP based data plane

- OpenFlow was designed for an ASIC based data plane
  - ‘misconception when SDN started’: ‘network comprised entirely of hardware switches’, citing Scott Shenker, Stanford Seminar - Software-Defined Networking at the Crossroads
- a GPP based data plane is naturally programmed directly in a high level programming language w/o emulating match-action pipelines
  - Linux kernel / hypervisor / container networking and other GPP based implementations like DPDK, FD.IO and Netmap are examples
  - vSwitch is the only counter example, but only used for rather simple L2 networking in IT clouds
- full Ethernet and IP networking remains THE fundamental requirement for a Telco cloud
  - no business case to re-invent/re-implement it with OpenFlow or P4 for GPP

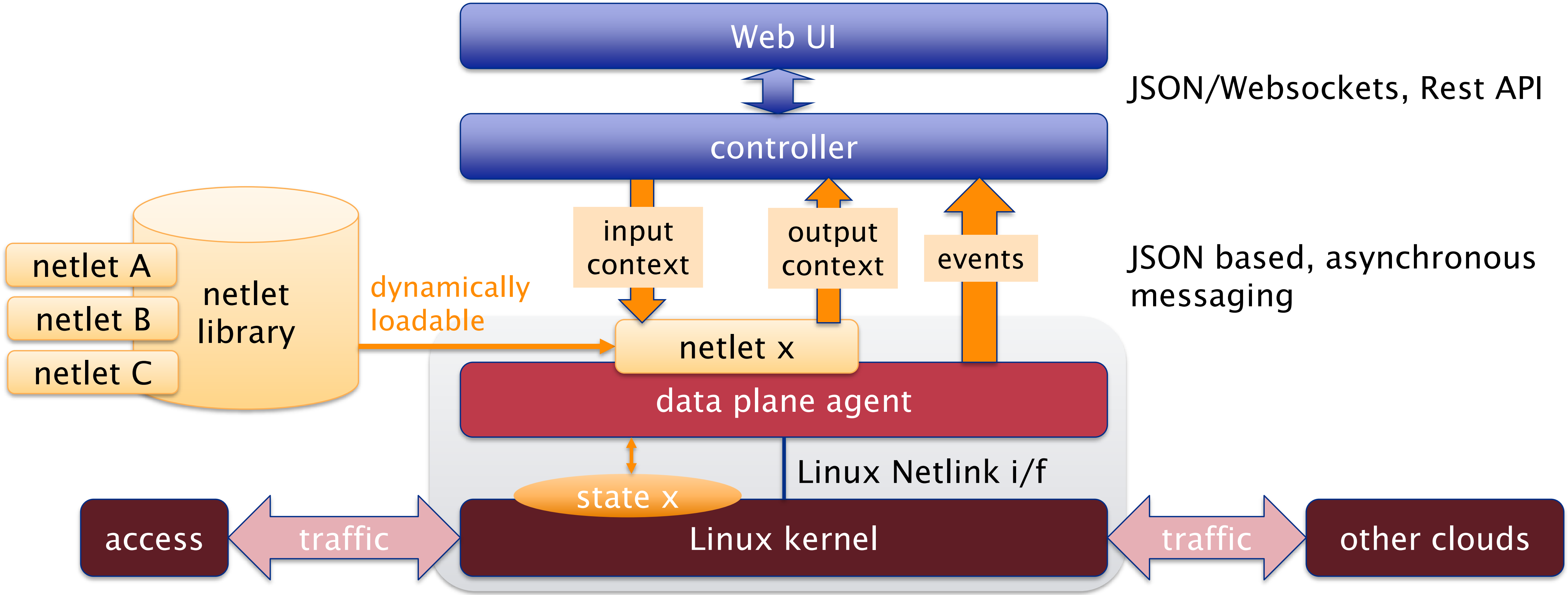
our proposed PoC: x86/Linux based service edge

investigating potential and realities with a Linux based prototype for a vRGW service

- towards an access agnostic, programmable IP/Ethernet edge built on commodity servers with open source software
- use the latest Linux kernel as a programmable data plane
  - basically following RFC 3549 (‘Linux Netlink as an IP Services Protocol’)
  - utilize recent improvements (eBPF, XDP)
- Proof-of-Concept: virtual residential gateway service (vRGW) on top of Linux servers
  - plumbing the capabilities and limits of open source networking
  - scalability, programmability, reliability, performance, stability, networking features
- cloud ready, distributed architecture
  - scale-out of control and data plane
  - fine-granular functional disaggregation
  - micro-service like approach for networking

what we did: Linux kernel as programmable data plane

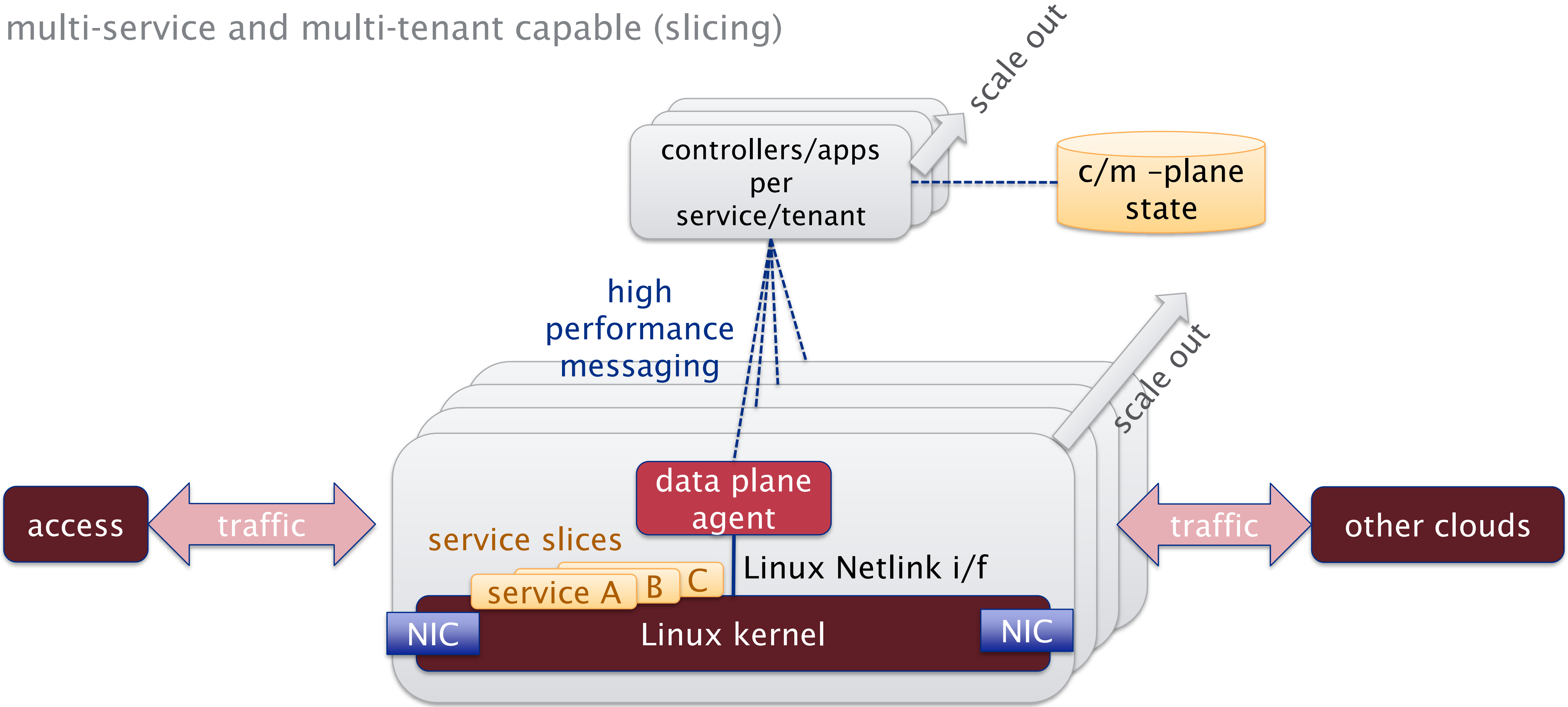
programming the data plane with networking abstractions of Linux



a netlet performs a generic ‘micro’ networking task as a transaction on top of the Linux kernel

independent scale-out of user and control plane

multi-service and multi-tenant capable (slicing)



summary of our preliminary results

a modern Linux kernel easily supports the vRGW functions

- main network functions used: network namespaces, nftables (NAT), dnsmasq (DHCP, DNS server), routing, bridging, ARP-proxy, unnumbered interfaces, GRE-tunnels for Ethernet and IP overlays
- created up to 500 vRGW instances on a single VMWare VM (HP ZBook G3, 4 logical cores of XEON E3-1505M, 2.8GHz, 24GB)
  - creation speed 3-5 vRGW instances per second
  - single instance requires ~20 MB main memory
  - so far no hard kernel limits hit
- traffic throughput
  - to be tested, however when engineered bandwidth moderate (< some Mbit/s/home) throughput unlikely to be the bottleneck