Skalierbare virtuelle Netz-Testbeds für Lehr- und Forschungsumgebungen mit VIRL

Sebastian Rieger [1]

Abstract: Lehrveranstaltungen und Forschungsprojekte, in denen Experimente im Netzwerkbereich durchgeführt werden, verwenden häufig virtuelle Netzwerkumgebungen. Diese können je nach erforderlicher Praxisnähe unterschiedliche Software-Lösungen bzw. Experimentierumgebungen verwenden. Eine hohe Praxisnähe in Bezug auf die Funktionalität des virtuellen Netzes bietet die Emulation von Netzen. Allerdings erfordert sie aufgrund der Nachbildung realer Netzfunktionen und -komponenten z.B. im Vergleich zu Simulationen weitaus mehr Ressourcen, was die Skalierbarkeit zugunsten der Praxisnähe erschwert. Das Paper beschreibt die im Umfeld des Netzwerk-Labors (NetLab) an der Hochschule Fulda eingesetzten Lösungen für emulierte virtuelle Netz-Testbeds und zeigt Möglichkeiten für die Bewertung von deren Leistungsfähigkeit und Skalierbarkeit auf.

Keywords: Netz-Virtualisierung, Simulation, Emulation, OpenStack, VIRL.

1 Einleitung

Für die Realisierung von experimentellen Netzwerkumgebungen für Forschungsprojekte und Lehrveranstaltungen haben sich unterschiedliche Möglichkeiten etabliert. Diese können anhand ihrer Ausrichtung auf Praxis oder Theorie differenziert werden. Abbildung 1 zeigt unterschiedliche Ausrichtungen und einige Beispiele für passende Werkzeuge.

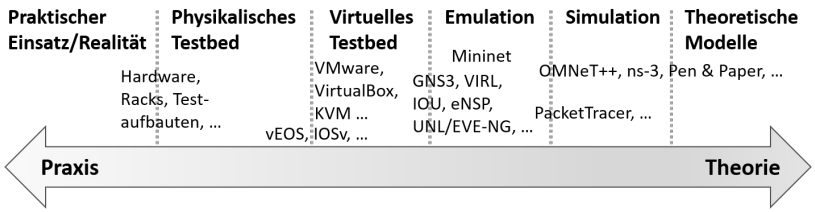


Abb. 1: Klassifizierung von Umgebungen für Netzwerkexperimente.

Auf der einen Seite können praxisnahe Testumgebungen, wie in der linken Seite der Abbildung gezeigt, direkt in realen Netz- und IT-Infrastrukturen realisiert werden. Dies ist allerdings aufgrund der Beeinflussung des Produktivbetriebs durch sowie auf das Testbed häufig nicht praktikabel. Um dieses Problem zu umgehen, können physikalische Testbeds isoliert vom Produktivsystem betrieben werden. In der Regel ist dies jedoch aufwändig und kann nur bedingt an schnell veränderliche Anforderungen angepasst werden. Virtuelle Testbeds reduzieren diesen Aufwand erheblich. Netzstrukturen (Links, Netzwerkkomponenten)

[1] Hochschule Fulda, Fachbereich Angewandte Informatik, Leipziger Straße 123, 36037 Fulda, sebastian.rieger@informatik.hs-fulda.de

müssen darin allerdings häufig manuell nachgebildet werden. Auf der anderen Seite können komplexe Netze, wie im rechten Teil der Abbildung 1 dargestellt, auch in rein theoretischen Modellen abgebildet werden. Eine Umsetzung der formalen Definition von Netzen und Erkenntnissen aus diesen Modellen in reale Umgebungen kann jedoch aufgrund der fehlenden Praxisnähe eine große Herausforderung bilden. Simulationen nähern sich der Praxis an und bieten den Vorteil, dass das Verhalten der zugrundeliegenden theoretischen Modelle exakt (z.B. bzgl. des Timings) auf Anforderungen aus der Praxis ausgerichtet werden kann. Zusätzlich kann innerhalb deterministischer Simulationen die Zeit im Gegensatz zu realitätsnäheren Ansätzen angehalten sowie vor- oder zurückgestellt werden. Simulationen verwenden ein abstraktes Netz, das das Verhalten in der Praxis nicht vollständig nachbilden kann. Entsprechend ist der Realitätsbezug abhängig von der konkreten Fragestellung, für die die Simulation entworfen wurde.

In [Ha12] wird basierend auf einer ähnlichen Klassifizierung eine Emulation statt Simulation der Netzkomponenten und Links empfohlen. Die Autoren des Papers bewerten dabei die Ziele bzw. Vorteile der Emulation von Netzumgebungen in den Bereichen: *Functional Realism*, *Timing Realism*, *Traffic Realism*, *Topology Flexibility*, *Easy Replication* und *Low Cost*. Emulatoren erfüllen gemäß der Bewertung in [Ha12] lediglich die Anforderung *Timing Realism* nicht ohne Weiteres exakt. Dies deckt sich auch mit den Erfahrungen aus den im Netzwerk-Labor (NetLab) an der Hochschule Fulda eingesetzten Lösungen. Entsprechend bildet die Emulation derzeit eine flexible Grundlage für die Realisierung von praxisnahen Netz-Experimentierumgebungen. Allerdings entsteht durch die realitätsnahen virtuellen Maschinen (VMs) im Vergleich zu Simulationen etc. ein hoher Ressourcenbedarf. Für große virtuelle Netztopologien sind zudem die lokalen Ressourcen von Laborrechnern schnell erschöpft. Dieses Paper zeigt eine Möglichkeit für die Verbesserung der Skalierbarkeit und die Verwendung einer zentral verwalteten Virtualisierungsplattform auf.

Die nächsten beiden Abschnitte stellen Beispiele für Netzwerkexperimentierumgebungen, die im NetLab verwendet werden, vor und treffen eine Auswahl für die in diesem Paper vorgestellte skalierbare Plattform für virtuelle Netz-Testbeds. Abschnitt 2 beschreibt die in diesem Paper verwendeten Netztopologien. Im Abschnitt 3 wird eine Testumgebung für die anschließend in Abschnitt 4 bewertete Skalierbarkeit von Netz-Testbeds unter Verwendung des Virtual Internet Routing Lab (VIRL) von Cisco realisiert. Abschließend zieht Abschnitt 5 ein Fazit und gibt einen Ausblick auf weitergehende Betrachtungen.

1.1 Virtuelle Umgebungen für Netzwerkexperimente

Die Emulation stellt gewissermaßen einen Mittelweg zwischen Theorie und Praxis dar und realisiert häufig ein virtuelles Testbed. So können reale Systeme (vgl. Linux VMs) und Netzwerkmanagementwerkzeuge (Wireshark etc.) innerhalb der Testbeds eingesetzt werden. Für Lehrveranstaltungen (insb. Übungen und Praktika) sowie Forschungsprojekte im Umfeld des NetLab werden *physikalische Testbeds*, *virtuelle Testbeds*, *Emulation* und *Simulation* eingesetzt. Analog zu den in [Ha12] diskutierten Ergebnissen hat sich dabei die Emulation als besonders flexibel herausgestellt. *Physikalische Testbeds* werden im NetLab z.B. für praxisnahe Projekte der Studierenden beispielsweise in Form von mobilen

Experimentier-Racks für studienbegleitende Cisco-Zertifizierungen (vgl. CCNA, CCNP) eingesetzt [PS11]. Sie bieten teilweise einen höheren *Functional Realism*, *Traffic Realism* und *Timing Realism*, jedoch zugunsten weitaus größerer Einschränkungen bei *Topology Flexibility*, *Easy Replication* und *Low Cost*. Für den Einsatz in Übungen, in denen die in der Vorlesung vermittelte Theorie veranschaulicht werden soll, sind komplexe Aufbauten mit *physikalischen Testbeds* häufig zu zeitintensiv. Auch die Realisierung von *virtuellen Testbeds* (z.B. dezentral mit VMware Workstation oder zentral mit VMware vSphere) erfordert einen hohen Aufwand bzgl. der Vorbereitung und Wartung der VMs und Netze. Sie werden daher im NetLab insb. für praxisnahe Client-/Server-Anwendungen sowie in isolierten Umgebungen z.B. für Experimente im Bereich der IT-Sicherheit eingesetzt. Die VMs müssen hierbei über die Semester hinweg überarbeitet (Updates, Änderungen am Setup bzw. der Netz- und IT-Infrastruktur) und verwaltet werden, was *Topology Flexibility* sowie *Easy Replication* einschränkt. Simulatoren wie ns-3 [ns17] und OMNeT++ [OM17] werden in Master-Veranstaltungen angesprochen. Für praxisnahe Übungen werden sie im NetLab jedoch neben ihrer Komplexität insb. aufgrund des fehlenden *Functional Realism* und *Traffic Realism* nicht verwendet.

Für die o.g. Cisco-Zertifikate wird PacketTracer [Pa17] eingesetzt. In Übungen zu Lehrveranstaltungen kam jedoch Kritik an dessen fehlender Praxisnähe auf. Beispielsweise werden Clients lediglich simuliert und bieten keine vollständige Implementierung gängiger Werkzeuge (z.B. reduzierter Funktionsumfang bei arp, ping, traceroute). Zusätzlich müssen Eigenheiten der Simulation erklärt werden. Beispielsweise verwirft PacketTracer das erste ICMP-Paket eines Pings am Router im Ziel-Subnetz, da dieser zunächst per ARP die MAC-Adresse des Ziel-Rechners ermitteln muss. Dieses für eine Simulation durchaus korrekte Verhalten tritt jedoch in der Praxis nicht auf, da der Client unmittelbar nach dem Boot-Vorgang bereits Pakete über den Router gesendet hat, und seine MAC-Adresse somit bereits im ARP-Cache des Routers liegt. PacketTracer zeigt daher Schwächen im Bereich *Functional Realism*, *Traffic Realism* und *Timing Realism*.

1.2 Begründung der Auswahl und Verwendung von VIRL

Im vorherigen Abschnitt wurde die Auswahl basierend auf den Anforderungen des NetLab bereits auf den Bereich „Emulation“ der Abbildung 1 begrenzt. Das z.B. in [LHM10] und [Ha12] in diesem Bereich verwendete Mininet [Mi17] wird im NetLab für SDN-Übungen eingesetzt. Aufgrund der Container-basierten Emulation besitzt es einen geringen Ressourcenbedarf. Allerdings erfordert die Realisierung von eigenen Netztopologien eine Einarbeitung in die Python-API von Mininet und es können nur bedingt reale Netz-Infrastrukturen/-Software integriert werden. Für eine praxisnahe Emulation von Netzumgebungen im NetLab wurden daher in den vergangen Jahren GNS3 [GN17], EVE-NG [Em17], eNSP [eN17] und VIRL [VI17] evaluiert. Alle setzen primär auf die Integration von VMs und erlauben so den virtuellen Einsatz realer Netz-Soft- und Hardware. eNSP ist auf Netzkomponenten von Huawei ausgerichtet. Die Clients sind analog zu PacketTracer simuliert und eingeschränkt. Wie auch GNS3 und EVE-NG kann eNSP allerdings lizenzkostenfrei verwendet werden. In Bezug auf die genannten Emulatoren besitzt GNS3 die größte Verbreitung und ist am längsten auf dem Markt, bietet jedoch kein cluster-basiertes scale-out und ist somit bzgl.

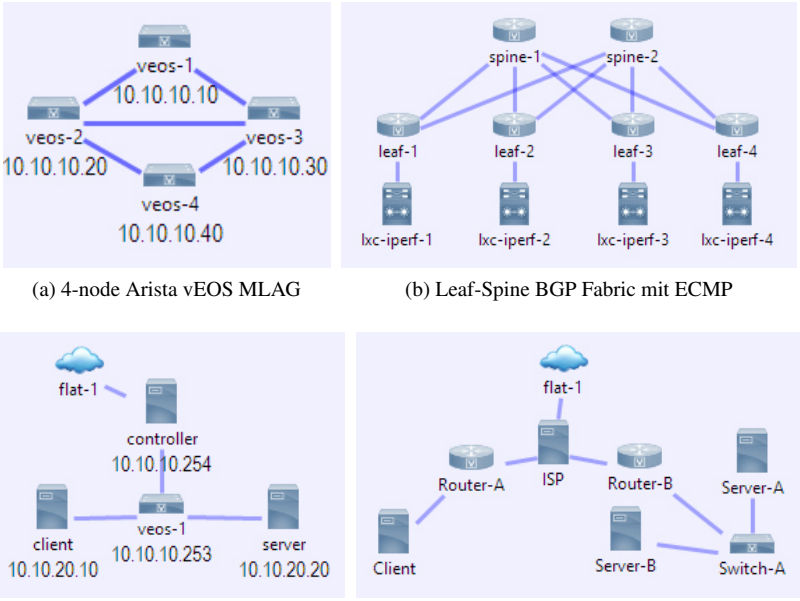
der Skalierbarkeit einzelner Topologien limitiert bzw. erlaubt keine Zusammenarbeit von Studierenden an laufenden Topologien. EVE-NG bietet hierfür eine Lösung und setzt darüber hinaus auf eine fortschrittliche web-basierte Konfiguration. Allerdings befindet sich EVE-NG noch im Alpha-Stadium. Alle bislang genannten Funktionen bietet auch VIRL. Aufgrund der auf OpenStack basierenden Architektur ist bei VIRL zusätzlich eine zentrale und skalierbare Cluster-Installation möglich. Emulierte Netztopologien können daher von den Studierenden gemeinsam und ortsunabhängig (im NetLab sowie auf dem eigenen Notebook bzw. von zu Hause) verwendet und mittels GIT versioniert verwaltet werden. Die offene Architektur von OpenStack und VIRL (auf der Basis von Ubuntu 14.04, LXC, linux-bridge, VXLAN) ermöglicht außerdem eigene Erweiterungen für die Realisierung von skalierbaren Netz-Testbeds für Lehr- und Forschungsumgebungen. Außerdem können Studierende im Vergleich zu den anderen Alternativen offizielle Cisco-Images nutzen, die bei GNS3 und EVE-NG separate Lizenzen erfordern.

VIRL erlaubt die direkte Verwendung von Betriebssystemen, die auch auf physikalischen Netzwerkkomponenten laufen (vgl. Arista EOS, Cisco IOS), in Form von VMs, was die im vorherigen Abschnitt genannte Anforderung *Functional Realism* nahezu vollständig erfüllt. Durch die virtuelle Vernetzung dieser VMs kann realer Traffic eingespielt (Capture) oder nachgebildet werden, was sowohl *Traffic Realism* als auch in Teilen *Timing Realism* bietet. Die VMs können dabei nach Bedarf flexibel zu virtuellen Netzwerktopologien zusammengeschaltet werden, um so im Vergleich zur Realisierung mit realer physikalischer Hardware eine bessere *Topology Flexibility* sowie *Easy Replication* zu erfüllen. Nachteilhaft sind in Bezug auf das Kriterium *Low Cost* im Vergleich zu den anderen genannten Alternativen die Kosten, die durch die Einzelplatz-Lizenzen von VIRL entstehen und maximal 20 Cisco-Nodes in laufenden Topologien erlauben. Trotz dieses Nachteils stellt VIRL derzeit in Bezug auf die anderen genannten Anforderungen eine praktikable Lösung für die im NetLab durchgeführten Experimente dar. Es erlaubt zusätzlich als einzige Lösung derzeit eine zentral verwaltete Umgebung bei der die Studierenden gemeinsam auf verteilt laufenden Topologien im Cluster arbeiten können. Der nächste Abschnitt zeigt einige Beispiele für die Realisierung von Topologien mit VIRL in Kombination mit praxisnahen Architekturen und Werkzeugen auf.

2 Beispiele für emulierte Netz-Testbeds und -Experimente im NetLab

Die Abbildung 2 zeigt einige Beispiele der im NetLab für Übungen und Praktika mit VIRL verwendeten Netztopologien. Abbildung 2a zeigt eine Topologie mit 4 Arista vEOS (4.16.9) Nodes. Die Links zwischen den vier Nodes sind jeweils redundant ausgelegt. Master-Studierende experimentieren im Team in unterschiedlichen Szenarien z.B. mit dem Einsatz des Multiple Spanning Tree Protocol (MSTP), Link Aggregation Control Protocol (LACP) und Multi-chassis Link Aggregation Group (MLAG), um die Auswirkungen des Spanning-Tree-Protokolls und dessen Erweiterungen auf die Nutzung der verfügbaren Links z.B. in Data Center Networks kennenzulernen. Aufbauend auf dieser Übung arbeiten die Master-Studierenden in der in Abbildung 2b gezeigten Topologie an Leaf-Spine-Architekturen mit BGP Fabrics, wie sie z.B. in Cloud-Rechenzentren von Facebook und Microsoft eingesetzt werden. Dabei kommen für die Endpunkte Ubuntu 14.04 Container (LXC)

zum Einsatz. Die Spine und Leaf Switches werden mit IOSv (15.6(2)T) umgesetzt. In der Vergangenheit wurden ähnliche Topologien mit vEOS (ebenfalls BGP) und NX-OSv (FabricPath) umgesetzt. Auf den LXC Containern können die Studierenden per SSH reguläre Linux-Kommandos verwenden. Die Nodes und Links können während einer laufenden Emulation gestoppt bzw. getrennt werden. Außerdem kann der Traffic auf jedem Link aufgezeichnet und mit Wireshark ausgewertet, sowie QoS-Parameter (Delay, Jitter, Packet Loss) auf den Links angepasst werden. Für SDN-Experimente mit OpenDaylight (als OpenFlow Controller) und Arista vEOS (als OpenFlow 1.3 Switch) wird die in Abbildung 2c gezeigte Topologie verwendet. Hierfür wurde zuvor eine Mininet-basierte Übung eingesetzt. Durch die Verwendung von VIRL können die Studierenden praxisnah unter Verwendung von vEOS, das sich analog zum EOS auf physikalischen Arista Switches verhält, den Einsatz von OpenFlow testen.



(a) 4-node Arista vEOS MLAG (b) Leaf-Spine BGP Fabric mit ECMP

(c) vEOS/OpenDaylight SDN Beispiel (d) IOSv/netem Routing/Switching Toubleshooting

Abb. 2: Beispiele für emulierte Netz-Testbeds und Netzwerkexperimente.

Ein Beispiel für die Verwendung weniger komplexer Netztopologien zeigt die Abbildung 2d. In dieser Übung ermitteln Bachelor-Studierende Fehlkonfigurationen (ARP, Routing, Delay/Packet Loss, Port Status/Shutdown), fertigen durch Verwendung realer Netzwerk-Tools (ping, traceroute/mtr, Wireshark) Netztopologien an und realisieren den Zugriff auf einen auf Server-B installierten Apache Web-Server. Client, Server und ISP bilden Ubuntu 14.04 VMs. Beim ISP wurde zusätzlich mit netem ein künstliches Delay und Packet Loss hinzugefügt (WAN Emulation). Der ISP Node bindet die Topologie als Standard-Gateway an das reale lokale Netz (flat-1) an. Damit ist ein Zugriff auf das Internet (ping auf google.com etc.) aus der Emulation möglich. Aus dem Labor können die Studierenden zusätzlich per OpenVPN auf Nodes im emulierten Netz (z.B. den Web-Server) zugreifen. Ein Zugriff außerhalb

des Labors (z.B. von zu Hause) ist über ein VPN-Profil der Hochschule möglich. Neben IOSv und Ubuntu Nodes kommt für den Switch IOSvL2 (15.2(4.0.55)E vgl. Catalysts) zum Einsatz. Alle genannten Beispiele werden per GIT unter [HS17a] bereitgestellt. Für den Einsatz von VIRL im NetLab wurden einige Erweiterungen implementiert. Diese umfassen z.B. eine an der Hochschule entwickelte Anpassung von Arista vEOS und CumulusVX Images für den Einsatz in VIRL [Ri17].

3 Realisierung einer VIRL Benchmark-Umgebung

Im Vergleich zu Simulationen, die in der Regel in wenigen Sekunden gestartet sind, benötigen die realitätsnahen virtuellen Testbeds in VIRL teilweise mehrere Minuten für den Start. Um diesen Nachteil zu reduzieren, wurde eine Cluster-Umgebung (VIRL 1.2.83) realisiert und evaluiert, um die Startzeit der Topologien zu optimieren. Da die Topologien direkt einsatzfähig konfiguriert sind und nach dem Start sofort laufen bzw. Daten übertragen, ist aus der Startzeit auch ein Rückschluss auf die Leistungsfähigkeit der Emulation während des Betriebs möglich. Die für die VIRL Umgebung verwendete Hardware wurde bereits in [PTR15] beschrieben. Als VIRL Hosts wurden hierbei vier VMs mit je 64 GB RAM und 32 vCPUs verwendet. Die VMs bilden eine Nested Virtualization innerhalb einer VMware vSphere 6.0 Umgebung, wobei jede der vier VIRL VMs durch eine DRS-Beschränkung explizit auf einem der vier physikalischen ESXi Hosts läuft. Die ESXi Hosts bieten 2 8-Kern Intel(R) Xeon(R) E5-2650v2 2.60GHz CPUs sowie 256GB RAM und nutzen als Storage-Backend zwei NetApp E2700. Die Knoten sind mit zweimal 1 GbE an einen Cisco C3850 und mit zweimal 10 GbE an einem Arista 7150S-24 und Arista 7050S-52 Switch angeschlossen. Wie auch in [Ha12] diskutiert wurde, hängt die Anforderung *Timing Realism* in Bezug auf die Emulation insb. in virtualisierten Umgebungen stark von der Isolation der Emulationsumgebung ab. Daher wurden für die nachfolgenden Benchmarks isolierte Resource Pools definiert und die Tests bewusst während der vorlesungsfreien Zeit durchgeführt sowie die Gesamtauslastung der VMware vSphere-Umgebung überwacht. Der VIRL Benchmark bildete zum jeweiligen Testzeitpunkt die einzigen VMs, die eine signifikante Last auf der vSphere-Umgebung erzeugten. Um zusätzlich etwaige Ausreißer in den Performance-Messungen für die in diesem Paper betrachtete Skalierbarkeit von VIRL zu reduzieren, wurde ein Skript entwickelt, das einen reproduzierbaren Test ermöglicht.

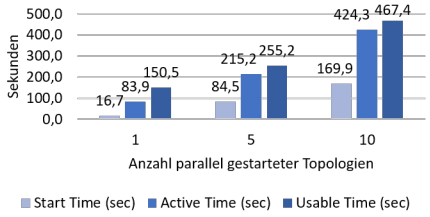
Für den Test wurde die in Abbildung 2a gezeigte Topologie verwendet, da sie über eine vergleichsweise kleine und damit gut skalierbare Node-Anzahl verfügt. Je Testzyklus werden die emulierten Netztopologien zunächst via VIRL REST-API gestartet, die Zeit bis zur Bestätigung des Starts gemessen und anschließend der Zeitpunkt ermittelt, zu dem alle Nodes *ACTIVE* waren. *ACTIVE* bedeutet in diesem Zusammenhang, dass die VMs vom OpenStack Nova Scheduler auf die VIRL Hosts verteilt wurden, die notwendigen virtuellen Netze und Ports angelegt wurden, das vEOS Image bereitgestellt ist und der Boot-Vorgang der VMs beginnt. Abschließend werden in dem entwickelten Benchmark-Skript Verbindungen zur Konsole der Nodes hergestellt und dadurch sowohl das interaktive Delay für Eingaben auf der Konsole als auch die Zeit gemessen, bis die Nodes tatsächlich durch die Benutzer verwendet werden können. Dafür wurde ein Python-Skript geschrieben, das eine WebSocket-Verbindung zur seriellen Konsole der Nodes auf den jeweiligen VIRL Hosts

herstellt, dreimal ENTER übermittelt und die Zeit bis zu den erwarteten resultierenden drei Prompt-Zeilen mit dem Hostnamen misst. Der geschilderte Testablauf wurde zunächst 10x nacheinander für jeweils eine emulierte Netztopologie gestartet. Dann wurden 10x nacheinander 5 parallel emulierte Netztopologien und schließlich 10x nacheinander 10 parallel Netztopologien gestartet und die o.g. Messungen bis zur vollständigen Nutzbarkeit der Topologien ermittelt. Jeder vEOS Node belegt 1 VCPU und 2 GB RAM. Somit waren für die Testzyklen mit 10 parallel gestarteten Topologien insg. 10 x 4 vEOS Nodes x 2 GB = 80 GB RAM erforderlich, die ab dem Boot-Vorgang sukzessive von KVM für die Prozesse reserviert wurden. Das Skript für den Benchmark und die dafür entwickelten Werkzeuge können unter [HS17b] abgerufen werden. Der komplette Testabauf wurde mit unterschiedlicher Cluster-Node-Anzahl wiederholt, um Rückschlüsse auf die Skalierbarkeit zu erhalten. Dabei wurde zunächst von einem einzelnen VIRL Host auf einen 2-Node Cluster und schließlich auf einen 4-Node VIRL Cluster erweitert. Der Testlauf wurde zu unterschiedlichen Tageszeiten wiederholt, um etwaige Seiteneffekte auf die Messungen zu minimieren. Die gemittelten durchschnittlichen Testergebnisse werden im nächsten Abschnitt präsentiert und ausgewertet.

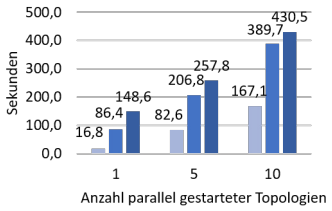
4 Bewertung der Skalierbarkeit

Abbildung 3 zeigt die Ergebnisse des im vorherigen Abschnitt erläuterten Testablaufs. Im Hinblick auf die Auswirkungen der Anzahl parallel gestarteter Topologien lässt sich z.B. in Abbildung 3a erkennen, dass die *Start Time* (Zeit bis alle per REST-API an VIRL übermittelten Topologien gestartet wurden) erwartungsgemäß linear ansteigt. Mit zunehmender Anzahl parallel gestarteter Topologien steigt jedoch die *Active Time* (Zeit bis alle VMs in den gestarteten Topologien booten) im Vergleich zur *Start Time* stärker an. Die *Usable Time* (Zeit bis die Konsole der gestarteten vEOS Nodes nutzbar ist) liegt mit zunehmender Node-Anzahl näher an der *Active Time*. Bis zu einer Anzahl von 20 parallel gestarteten Topologien sinkt der Zuwachs von *Active Time* und *Usable Time* zunächst. Zusätzliche Cluster Nodes steigern diesen Effekt, sofern die Anzahl parallel gestarteter Topologien größer als 5 ist. Bei mehr als 20 parallel gestarteten Topologien (vgl. Abbildung 3c) steigt die Wartezeit bis zur vollständigen Verfügbarkeit der Topologien allerdings bedingt durch die komplette Auslastung der Ressourcen des 4-Node Clusters wieder an. Wie in Abbildung 3c erkennbar ist, nimmt der Zuwachs an Wartezeit auf 1 oder 5 Topologien erwartungsgemäß durch den Aufwand für die Verteilung und Anbindung der zusätzlichen Cluster Nodes leicht zu. Diese Effekte lassen auf die Skalierbarkeit des OpenStack Schedulers schließen. Mit zunehmender Anzahl parallel gestarteter Nodes sinkt der Overhead durch das Scheduling. Neben dem Overhead durch das Scheduling resultieren die vergleichsweise langen Wartezeiten auf das Starten der VMs (Abstand zwischen *Start Time* und *Active Time*) auch aus dem aufwändigen Prozess der Anlage von virtuellen Netzen. Jeder Link in Abbildung 2a ist redundant und erfordert die Anlage von zwei VXLAN-Segmenten und dazugehörigen Ports in den Linux Bridges etc. Dies äußert sich auch in einer hohen Auslastung der Neutron Prozesse auf dem Controller Node des OpenStack Clusters. Testweise wurde die Anzahl der neutron-server und nova-api sowie nova-conductor Worker Prozesse auf 10 erhöht. Aufgrund der hierfür zusätzlich erforderlichen Ressourcen konnten

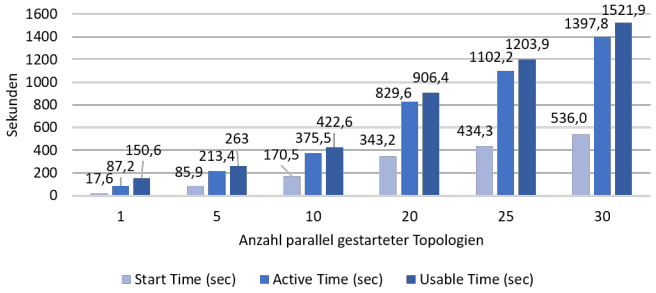
danach nur noch maximal 20 Topologien gestartet werden. Entsprechend wurde der auf dem Controller des VIRL Clusters benötigte Arbeitsspeicher reserviert. Dies reduziert die Wartezeit auf die 20 parallel gestarteten Topologien gemäß Abbildung 3d jedoch nur um ca. 11%. Hier zeigt sich noch Verbesserungspotenzial bzgl. der Anpassung der Konfiguration des OpenStack Managements (Scheduler, Message Bus) sowie dem nachgelagerten KVM.



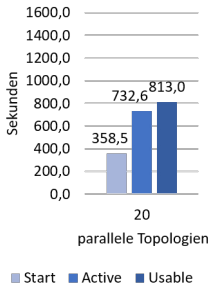
(a) Startzeit Topologie Abb. 2 (a) - 2-Node Cluster



(b) 4-Node Cluster mit ramdisk



(c) Startzeit Topologie Abb. 2 (a) - 4-Node Cluster



(d) 4-Node - mehr Worker

Abb. 3: Benchmark-Ergebnisse für die Startzeit der emulierten Netztopologien.

Limitierende Faktoren bilden in Bezug auf den Benchmark eher RAM und I/O als die CPU-Leistung. CPU-Last entsteht neben den OpenStack- und VIRL-Management-Prozessen primär beim Booten der vEOS Instanzen, wobei die zugewiesene VCPU für durchschnittlich 60 Sekunden zu 100% ausgelastet ist. Von Cisco wird für VIRL die Option angeboten eine ramdisk für die Instanzen (Nova VMs) zu nutzen. Die Abbildung 3b zeigt den testweisen Einsatz einer ramdisk. Da dies von Cisco nur für den Controller unterstützt wird, wurden die Compute Nodes manuell angepasst. vEOS Instanzen nutzen allerdings nur jeweils 213 MB ephemeral storage. Daher ist der positive Effekt einer ramdisk gering bzw. wird bei zunehmender Anzahl paralleler Topologien durch den Performance-Verlust durch das zusätzlich belegte RAM (vgl. Abbildung 3b) zunichte gemacht.

5 Fazit und Ausblick

Durch Cisco VIRL wird für Lehrveranstaltungen wie insb. Praktika und Übungen sowie für Forschungsprojekte eine praxisnahe und skalierbare Plattform für virtuelle Netz-Testbeds bereitgestellt. Gegenüber Alternativen wie GNS3 und EVE-NG bietet VIRL den Vorteil, dass offizielle Cisco-Images ohne zusätzliche Lizenzen verwendet werden können. Der

weitaus wichtigere Vorteil gegenüber den Alternativen ist jedoch die OpenStack-basierte Cluster-Lösung von VIRL, die mit einigen manuellen Anpassungen, wie im Paper präsentiert, eine Skalierbarkeit auch für große Topologien sowie eine zentral verwaltete Lösung für große Benutzergruppen ermöglicht. Für VIRL Hosts wird so ein scale-out für virtuelle Testbeds möglich. Durch die Verwendbarkeit von Standard-Netzwerk-Management-Werkzeugen (vgl. Wireshark, ping, traceroute, usw.) bzw. Ubuntu Containern und VMs in den emulierten Netztopologien sowie die Anbindung an reale externe Netze und das Internet wird im Vergleich zu Netz-Simulatoren eine hohe Flexibilität und Realitätsnähe erreicht. Die damit verbundene im Vergleich zu Netz-Simulatoren längere Startzeit, wird durch einen für das NetLab in PHP und Python entwickelten VIRL-Scheduler zusätzlich kompensiert. Dieser wird verwendet, um zeitgesteuert vor einer Übung Topologien für die Studierenden zu laden und so eine direkte Verwendung in der Lehrveranstaltung zu erlauben. In einem Bachelor-Projekt ist zudem derzeit von mehreren Studierenden ein VIRL-Kursmanagement in Entwicklung. Dies soll neben dem VIRL-Scheduler auch die automatisierte Anlage von Kursen (Benutzern und Gruppen) für VIRL sowie ein Reservierungssystem für automatisierte Starts von Topologien außerhalb der Zeiten der Lehrveranstaltungen erlauben. Das Dev/Innovate/Research Program, über das die Hochschule Fulda die Cluster-Lizenz für dieses Paper erhalten hat, soll 2017 auslaufen. Derzeit wird evaluiert auf die offizielle Mehrplatz-Version von VIRL (Cisco Modeling Labs (CML)) umzustellen, oder eine Einzellizenz für den Cluster zu verwenden. Die dadurch entstehende Limitierung auf max. 20 Cisco-Nodes wäre im Labor vertretbar, zumal andere VMs (z.B. Arista, Cumulus, Linux) trotzdem unbeschränkt verwendet werden können. Die aktuelle Entwicklung bei GNS3 und EVE-NG läuft ebenfalls in Richtung skalierbarer Cluster-Lösungen, in denen Topologien gemeinsam von mehreren Nutzern verwendet werden können, so dass hierbei ggf. eine kostengünstige Alternative zu VIRL für das NetLab entsteht.

Bzgl. weiteren Performance-Steigerungen sind Optimierungen des OpenStack Scheduling und der Netz-Infrastruktur, wie in der Bewertung in diesem Paper angesprochen, denkbar. Im Bereich SDN bleibt Mininet durch die extrem schnelle Startzeit von Switches und Containern, trotz geringerer Praxisnähe, eine interessante Alternative. Gleiches gilt für einige Übungen, in denen virtuelle Testbeds basierend auf lokalen VMware Workstation/VirtualBox Installationen oder physikalische Racks aufgrund einer geringen Komplexität der Netzexperimente besser einsetzbar sind. Zukünftig ist durch neue OpenStack Versionen mit einer Steigerung der Performance des Scheduling (nova-scheduler) sowie der Prozesse für die Verwaltung der virtuellen Netz-Infrastruktur (neutron-server) zu rechnen. Derzeit werden diese in VIRL nur mit jeweils einem Prozess ausgeführt und limitieren daher die Startzeit der emulierten Netztopologien. Zudem wird nur ein sequentielles Scheduling ermöglicht, was einen hohen Traffic auf der verwendeten Message Queue und Datenbank sowie den Log-Files etc. zur Folge hat. Dies bietet ebenfalls Ansatzpunkte für zukünftig im Umfeld des NetLab betrachtete Verbesserungen.

6 Danksagung

Für das NetLab wurde von Cisco eine VIRL Lizenz im Rahmen des Dev/Innovate/Research Program zur Verfügung gestellt.

Literaturverzeichnis

[Em17] Emulated Virtual Environment Next Generation (EVE-NG) / Unified Networking Lab (UNL), http://www.unetlab.com, Stand: 3.1.2017.

[eN17] eNSP - Enterprise Network Simulator, http://support.huawei.com/enterprise/en/network-management/ensp-pid-9017384, Stand: 3.1.2017.

[GN17] GNS3 - The software that empowers network professionals, https://www.gns3.com, Stand: 3.1.2017.

[Ha12] Handigol, Nikhil; Heller, Brandon; Jeyakumar, Vimalkumar; Lantz, Bob; McKeown, Nick: Reproducible network experiments using container-based emulation. In: Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, S. 253–264, 2012.

[HS17a] HS-Fulda NetLab VIRL Topologien, https://gogs.informatik.hs-fulda.de/srieger/git-virl-hs-fulda, Stand: 3.1.2017.

[HS17b] HS-Fulda NetLab VIRL Utilities, https://gogs.informatik.hs-fulda.de/srieger/virl-utils-hs-fulda, Stand: 3.1.2017.

[LHM10] Lantz, Bob; Heller, Brandon; McKeown, Nick: A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, S. 19, 2010.

[Mi17] Mininet - An Instant Virtual Network on your Laptop (or other PC), http://mininet.org, Stand: 3.1.2017.

[ns17] ns-3, https://www.nsnam.org, Stand: 3.1.2017.

[OM17] OMNeT++ Discrete Event Simulator, https://omnetpp.org, Stand: 3.1.2017.

[Pa17] Packet Tracer - A free network simulation and visualization tool for the IoT era., https://www.netacad.com/about-networking-academy/packet-tracer, Stand: 3.1.2017.

[PS11] Pape, Christian; Seifert, Christoph: Adaption and improvement of an industry-developed IP Telephony curriculum. In: 7th Annual International Conference on Computer Science and Education in Computer Science. Sofia/Dobrinishte, Juli 2011.

[PTR15] Pape, C.; Trommer, R.; Rieger, S.: Energieverbrauch von Live-Migrationen in OpenStack-basierten Private-Cloud-Umgebungen. In: 8. DFN-Forum - Kommunikationstechnologien, 8.-9. Juni 2015, Lübeck, Germany. 2015.

[Ri17] Arista vEOS image on VIRL, https://learningnetwork.cisco.com/thread/99040, Stand: 3.1.2017.

[VI17] VIRL - Virtual Internet Routing Lab, http://virl.cisco.com, Stand: 3.1.2017.