# Das Baltikum Testbed

Christian Feiler
Betreuer: Daniel Raumer
Seminar Future Internet SS2016
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: christian.feiler@tum.de

## ABSTRACT
Das Durchführen von Experimenten ist ein wichtiges Hilfsmittel beim Testen neuer Services oder Ermitteln von Daten. Um Experimente reproduzierbar ausführen zu können, wird eine spezielle Umgebung benötigt. Das Baltikum Testbed der Technischen Universität München bietet eine solche Umgebung. Schwerpunkte dieser Arbeit sind vor allem auf den Aufbau des Testbeds und den Ablauf von Experimenten gelegt. Weiterhin werden die Möglichkeiten des Testbeds aufgezeigt und bisherige Einsatzgebiete erläutert. Zusätzlich werden verschiedene Nutzungs- und Nutzerstatistiken für das Testbed in Anlehnung an andere Testbeds vorgestellt.

## Keywords
Testbed, Experiment, Statusseite, Statistiken

## 1. EINLEITUNG
Bei vielen neuen Services und Netzwerktechnologien sind Tests notwendig, um sämtliche Features zu analysieren. Bei diesen Tests ist es eine Erleichterung, wenn sie durch eine Umgebung kontrolliert ausgeführt werden können, welche es auch erlaubt, Experimente unter den gleichen Bedingungen beliebig oft auszuführen. Das Baltikum Testbed ist ein Beispiel für eine solche Umgebung, die auch komplizierte Experimente – wie sie zum Beispiel bei Multiprozessor-Anwendungen zustande kommen – realisieren kann. In über 15 Publikationen wurden Messwerte dieses Testbeds zitiert [6]. Im Gegensatz zu Simulatoren besteht ein Testbed aus realer Hardware und liefert somit authentische und realistische Ergebnisse. In Testbeds können eine Vielzahl an Messdaten erfasst werden, die zur Analyse von Services benötigt werden – unter anderem die Datenrate zwischen Netzwerkkomponenten oder die Ressourcen-Auslastung der Komponenten selbst durch den Service. Wie es im Baltikum Testbed ermöglicht wird, solche Daten zu erfassen, wird in Kapitel 2 erläutert, wobei sich über 30 studentische Arbeiten mit der Implementierung und Verbesserung der Funktionalität beschäftigen [6].

## 2. TESTBED ÜBERSICHT
Bevor der Ablauf von Experimenten und die verwendeten Technologien erläutert werden, ist es notwendig, sich mit dem Aufbau des Baltikum Testbeds vertraut zu machen.

### 2.1 Aufbau
Für die Ausführung von Tests stehen im Baltikum Testbed zehn Hosts mit verschiedenen Hardware Komponenten zur Verfügung. Wie dem passwortgeschütztem Wiki des Testbeds [3] entnommen werden kann, verfügen die Hosts jeweils über mindestens 16 GB Arbeitsspeicher und über eine CPU der Intel Xeon Reihe mit mindestens vier Prozessorkernen. Die Netzwerkkarten variieren je nach Host zwischen Kupfer- und Glasfaseranschluß und unterstützen mindestens 1 GBit/s – bei den meisten Hosts sind es 10 GBit/s. Somit kann eine Vielzahl an Experiment-Szenarien abgedeckt werden. Zusätzlich besteht für die Rechner die Möglichkeit, den Energieverbrauch während eines Experiments aufzuzeichnen. Daher können nicht nur Daten wie zum Beispiel CPU Auslastung, Paketrate oder Latency, sondern auch der Stromverbrauch einzelner Softwareabläufe gemessen werden. Die Messung des Energieverbrauchs erfolgt dabei über ein zentrales Messinstrument, das für jeden Host die Stromaufnahme misst [2]. Da Virtualisierung eine beliebte Rolle in der IT einnimmt [11], beschränken sich Experimente nicht nur auf die zehn physikalischen Hosts: Es können auf jedem Host Virtuelle Maschinen definiert werden. Außerdem stehen einem Benutzer des Testbeds verschiedene Betriebssysteme wie Linux, Windows oder FreeBSD für die Rechner zur Verfügung.
Neben der zehn Server und eines konfigurierbaren Switches für die flexible Verbindung der Hosts beinhaltet das Baltikum Testbed noch einen weiteren Host, der für Management Aufgaben zuständig ist. Auf diesem zentralen Host werden Experimente von Benutzern definiert, gestartet und im Anschluß die Ergebnisse eines Experiments abgelegt.
Die Hosts sind über mehrere Testnetzwerke untereinander und über ein Managementnetzwerk zentral verbunden.

### 2.2 Ablauf eines Experiments
Der Management-Rechner spielt eine zentrale Rolle bei Experimenten. Denn um ein neues Experiment zu definieren, muss auf diesem ein Konfiguration mit allen nötigen Informationen erstellt werden. Wichtige Inhalte könnten folgende sein:

- Konfiguration der (physikalischen) Hosts:
  - Netzwerkinterfaces
  - Betriebssystem Image
  - Skript
- Konfiguration der virtuellen Hosts/Maschinen: analog zu den physikalischen Hosts


doi: 10.2313/NET-2016-09-1_03

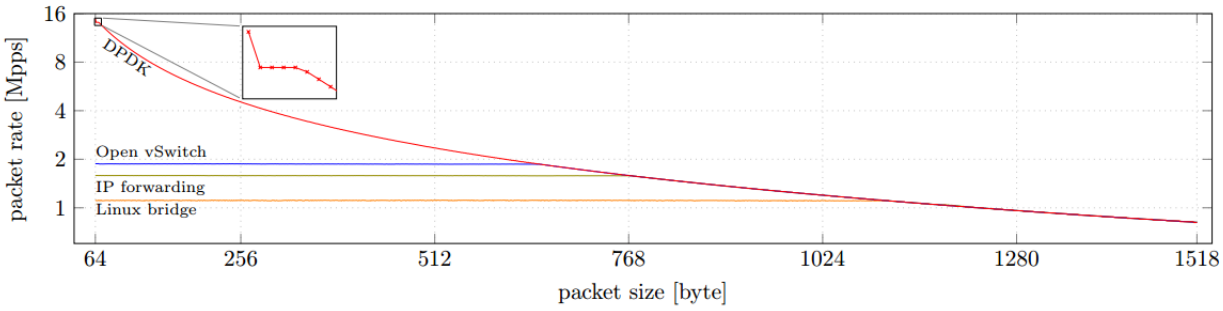


**Figure 1: Vergleich der Datenrate verschiedener Frameworks. Aus [10]**

- Konfiguration des Switches

Das Skript für die jeweiligen Hosts ist dabei ein Bash-Skript, das den Instruktionsablauf eines Hosts während eines Experiments beinhaltet. Für jeden verwendeten Rechner wird somit ein Skript benötigt. Um einen reproduzierbaren und synchronisierten Experimentablauf zu garantieren, gibt es die Möglichkeit, in diesen Experiment-Skripten Synchronisationspunkte einzubauen, welche durch Namen unterschieden werden. Wird ein Synchronisationspunkt erreicht, fährt der Host erst mit der Ausführung seines Skripts fort, wenn jeder andere Host den Synchronisationspunkt erreicht hat. Die Synchronisation erfolgt dabei über das Managementnetzwerk durch Anfragen an den zentralen Server.
Hat man nun die Konfiguration erstellt, kann man das Experiment starten, was folgende Schritte auf dem Managementhost als Resultat hat [3]:

- Die Konfigurationsdatei wird eingelesen.
- Es wird eine iPXE Konfiguration für die Testrechner angelegt. iPXE ist dabei ein Bootloader, der auf den Hosts installiert ist und die Möglichkeit schafft, das Betriebssystem als Image über ein Netzwerk zu beziehen.
- Es werden verschiedene Konfigurationsdateien und Skripte erzeugt, die die involvierten Hosts vor und während des Experiments über einen Apache Webserver vom Managementhost downloaden und für die Ausführung benötigen.

Nach diesen initialen Schritten erfolgt die Kommunikation über das Managementnetzwerk, wobei die Aktivitäten durch den Manangementserver angestoßen werden:

- Der Switch wird bei Bedarf mittels SNMP konfiguriert.
- Den Hosts wird mithilfe IPMI mitgeteilt, über das Netzwerk mittels iPXE zu booten. IPMI ist eine Schnittstelle, die das Fernsteuern von Rechnern auch im ausgeschalteten Zustand erlaubt.
- Die Hosts bekommen über DHCP eine IP Adresse für das Interface im Managementnetzwerk zugewiesen und laden anschließend mithilfe der vorher erstellten iPXE Konfiguration das Betriebssystem.
- Nachdem das Betriebssystem gestartet ist, laden die Test-Hosts die erzeugten Skripte – unter anderem das Experiment-Skript – und führen das Experiment-Skript aus.

Sobald ein Rechner mit der Ausführung seines Skripts fertig ist, meldet er dies dem Manangementserver mithilfe des Apache Webservers. Während des Experiments werden Dateien mit detailierten Informationen und Ergebnissen von den Hosts auf den zentralen Server hochgeladen. Wie viele Dateien geuploadet werden und was in den Dateien genau enthalten ist, muss im jeweiligen Skripte bestimmt werden.

## 3. EXPERIMENTSCHWERPUNKTE

Nachdem das Testbed an sich bekannt ist, stellt sich die Frage, für was das Testbed konkret verwendet wird. In der Einleitung dieser Arbeit wurde das Testen neuer Netzwerktechnologien als Beispiel genannt. Eine dieser Technologien sind Software Defined Networks (SDN's). Bei diesen Netzwerken sind die Control- und die Data-Ebene nicht mehr aneinander gebunden im Vergleich zu traditionellen Netzwerken. In diesem Fall schaffen Experimente eine geeignete Grundlage für den Performance-Vergleich der Netzwerke. Eine konkrete Implementierung von OpenFlow Switchen, die Einsatz bei SDN's findet und auch im Baltikum Testbed ausgiebig getestet wird [9], ist Open vSwitch. OpenFlow ist dabei ein Kommunikationsprotokoll, das eine Schnittstelle zwischen Control- und Data-Ebene ermöglicht.
Das Baltikum Testbed wird aber nicht nur zum Analysieren bereits entwickelter Technologien verwendet, sondern auch zum Erstellen neuer Services. Ein Beispiel hierfür ist der Packet Generator MoonGen [8]. Der Generator verwendet dabei das Network Framework DPDK, welches für schnelle Paket Verarbeitung optimiert ist.
Im Zusammenhang mit DPDK ist es auch interessant, dieses Framework mit anderen Frameworks für Paketweiterleitung zu vergleichen, da vermehrt auch handelsübliche Hardware statt Spezialhardware für den Einsatz als Netzwerkgerät verwendet wird [10], man aber möglichst ähnliche Performance erzielen will. Abbildung 1 zeigt als Beispiel das Ergebnis mehrerer Experimente, in denen Linux bridge, IP forwarding, Open vSwitch und Intel DPDK als Frameworks für Paket-Weiterleitung verglichen werden.
Für dieselben Experimente zeigt Abbildung 2 die zugehörige CPU Last Verteilung.
In Kapitel 2.1 wurde bereits angesprochen, dass auf den


doi: 10.2313/NET-2016-09-1_03

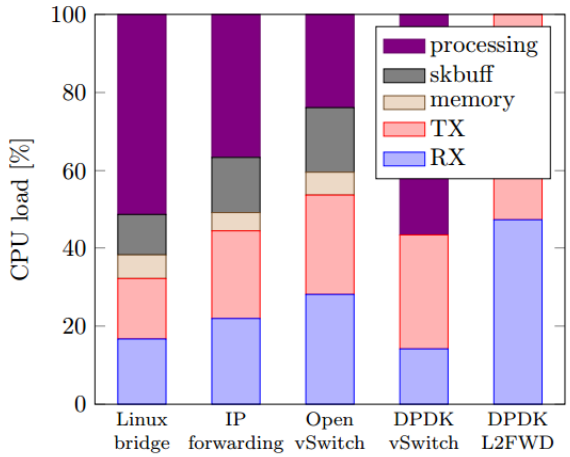


**Figure 2: Vergleich der CPU Lastverteilung verschiedener Frameworks. Aus [10]**

Hosts im Testbed verschieden Betriebssysteme verwendet werden können. Dadurch lassen sich viele Vergleiche der Betriebssysteme durch Experimente erstellen, wie zum Beispiel der Vergleich der generelle Netzwerkleistung auch in Hinblick auf Energieverbrauch [1].

# 4. NUTZUNGSSTATISTIKEN

Teil dieser Arbeit soll es auch sein, eine Statusseite für das Testbed mit aussagekräftigen Statistiken zu erstellen, mit denen sich die Nutzung des Testbeds analysieren lässt. Dazu ist es hilfreich, die Features der Statusseiten anderer Testbeds zu analysieren.

## 4.1 Statistiken anderer Testbeds

Die drei Testbeds, deren Statistikseiten im Weiteren erläutert werden, beinhalten das Emulab, das PlanetLab Europe und das FIT IoT-Lab. Diese Testbeds sind wesentlich größer dimensionert als das Baltikum Testbed, da sie zwischen 300 [7] und 2700 [5] Nodes beinhalten. Im Unterschied zum Emulab und zum PlanetLab Europe besteht das FIT Iot-Lab nicht aus normalen Servern, sondern aus Wireless Sensor Nodes, da es ein Internet of Things Testbed ist. Die unterschiedlichen Größen und Einsatzgebiete der Testbeds haben jedoch wenig Einfluß auf die Gestaltung der Statistiken, sodass die drei Statistikseiten ohne Bedenken als Orientierung für eine eigene Seite hergenommen werden können.

### *4.1.1 Emulab*

Die Statusseite des Emulabs [4] beinhaltet sechs Verlaufsdiagramme, die einen Überblick über die Zahl der verfügbaren Nodes schaffen. Die Diagramme zeigen sowohl den kurzfristigen Verlauf als auch den langfristigen Trend der letzten zehn Jahre.
Zusätzlich gibt es auf der Emulab-Website noch eine Liste über aktuell laufende und kürzlich abgeschloßene Experimente. Detaillierte Informationen zu den Experimenten gibt es leider nicht.
Außerdem kann man auf der Emulab Website eine Weltkarte finden, auf der abgelesen werden kann, wie viele Nutzer des Testbeds es in welcher Stadt gibt. Zwar macht diese Statistik wegen der geringen Größe wenig Sinn für das Baltikum Testbed, dennoch schafft sie einen interessanten Einblick in das Emulab.

### *4.1.2 PlanetLab Europe*

Im Vergleich zur Emulab-Statusseite beschränken sich die Statistiken nicht nur auf den Verlauf der verfügbaren Nodes, sondern geben einen breiten Einblick in die Nutzung des Labors. Zum einen gibt es unter anderem ein Verlaufsdiagramm zur Anzahl der Host-Standorte, der Nodes [7] und der User. Zum anderen gibt es auch Diagramme zum aktuellen Status der Nodes, der Verteilung der CPU-Architekturen oder der verwendeten Betriebssysteme.
Die Statistiken sind jedoch nur beschriftet und werden nicht beschrieben, sodass oftmals nicht ersichtlich ist, was ein Diagramm genau aussagt. Es gäbe nämlich auch Statistiken zur Auslastung der Nodes, jedoch ist es nicht erkennbar, ob es sich um einen Durchschnittswert handelt und welche Nodes für die Statistiken herangezogen wurden.

### *4.1.3 FIT IoT-Lab*

Im Vergleich zum Emulab und PLanetLab besteht das FIT IoT-Lab aus wesentlich mehr Nodes – aus über 2700 Wireless Sensor Nodes [5].
Dennoch ist auf der Website [5] im Reiter „Activity“ unter anderem ein Gantt-Diagramm abrufbar, welches detaillierte Informationen zu jedem Gerät liefert: Wann war der Node mit welchem Experimenten beschäftigt? Außerdem sind Informationen über aktuell laufende Experimente und über den Status der Nodes verfügbar.
Es ist jedoch auffällig, dass der Aufruf von Statistiken teilweise mit langen Wartezeiten (über 20 Sekunden) verbunden ist. Zusätzlich sind die Informationen weniger übersichtlich aufbereitet als bei den anderen beiden Testbeds, da keine Verlaufsdiagramme verwendet werden.

## 4.2 Baltikum Testbed Statusseite

Ziel dieser Arbeit ist es, die Vorteile der oben beschriebenen Statusseiten zu verbinden und die negativen Auffälligkeiten zu umgehen. Dazu sollen auf der Statusseite sowohl Live-Informationen wie der Status der Nodes (siehe IoT-Lab) und Verlaufsstatistiken über die Nutzung (siehe Emulab) angezeigt werden. Zwar ist es schwierig Statistiken zur Auslastung der Nodes zu erstellen wie beim PlanetLab Europe, jedoch soll es auf der Baltikum Testbed Statusseite einen Graphen mit der aktuellen Netzwerktopologie geben. Aufgrund der geringen Größe des Testbeds kann das übersichtlich realisiert werden. Um die Statistiken zu realisieren, werden im Backend Benutzerdaten und Experimentdaten gesammelt, die durch das Webfrontend mithilfe einer REST-API erfragt und dann visualisiert werden.
Außerdem sollen die Statistiken selbsterklärend oder beschrieben sein und nicht lange (unter fünf Sekunden) zum Laden brauchen.

### *4.2.1 Webserver*

Das Backend ist in Python mithilfe des Web Frameworks Pyramid realisiert und muss auf dem Managementhost ausgeführt werden. Der Server muss hierbei zum einen wie für einen Webserver typisch statische Dateien übertragen, damit der User das Webfrontend laden kann, zum anderen muss er spezielle HTTP-Requests erkennen, die mit Statusinforma-

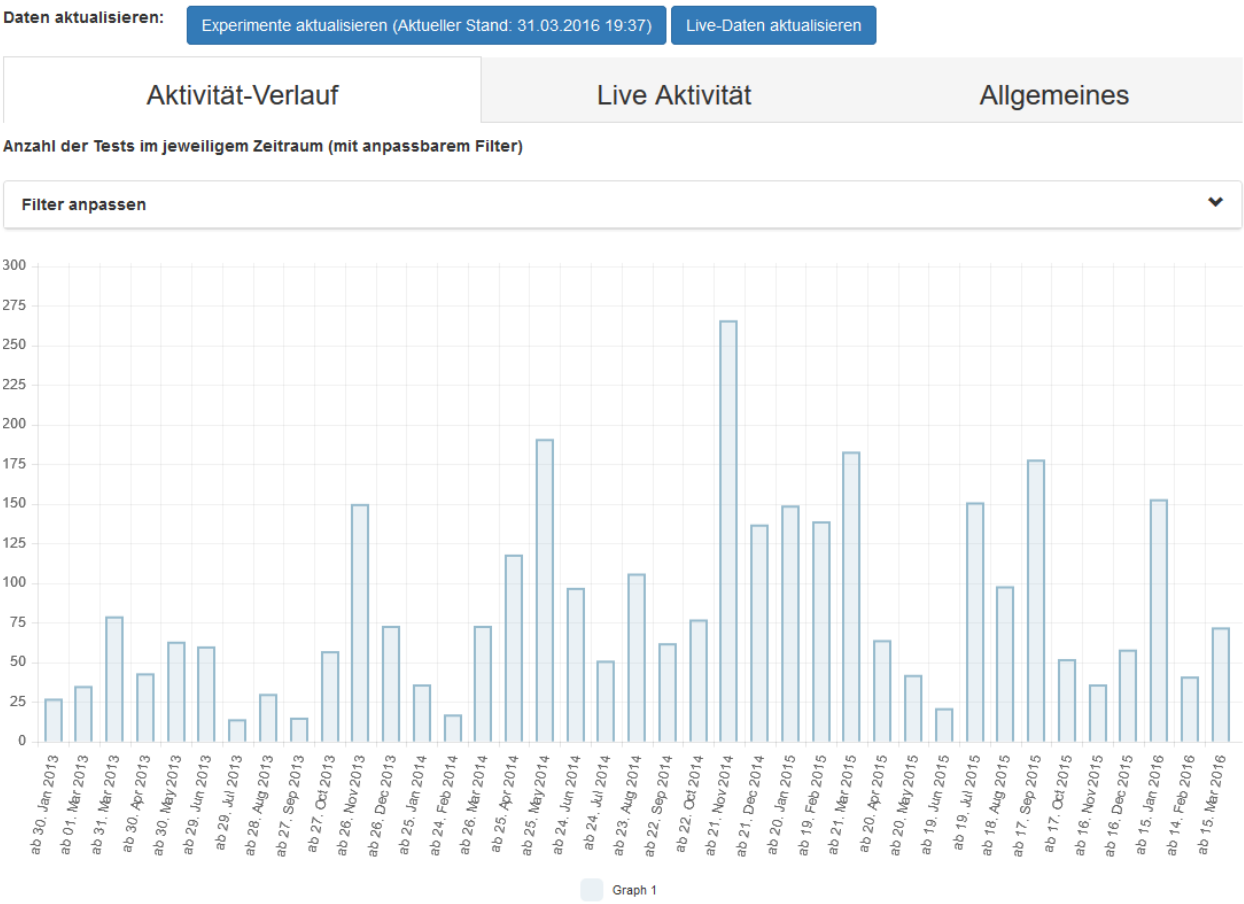


**Figure 3: Die Startseite der Statusübersicht des Baltikum Testbeds. Die Daten können durch die blauen Buttons aktualisiert werden.**

tionen beantwortet werden müssen. Diese HTTP-Requests sind folgende:

- `/api/last_logins`
  Der Output des Shell-Befehls `last` auf dem Managementserver wird zurückgesendet. Die Antwort beinhaltet also die letzten Logins auf dem Management-Host des Testbeds.
- `/api/logged_in_users`
  Als Antwort werden die Benutzernamen der User gesendet, die aktuell auf dem Testbed eingeloggt sind.
- `/api/host_status`
  Das Ergebnis dieser Anfrage beinhaltet, welche Hosts gerade eingeschaltet sind und ob sie im Moment an einem Experiment beteiligt sind.
- `/api/tests_sorted_by_date`
  Die Antwort beinhaltet alle Experimente als JSON-Objekt. Die Experimente sind dabei nach dem Datum und der Zeit sortiert, an dem sie abgeschlossen wurden. Zusätzlich werden zu jedem Experiment neben dem Datum und dem Namen noch die beteiligten Hosts und der Nutzer, der das Experiment ausgeführt hat, mitgesendet. Alle diese Informationen wurden zu einem vorherigen Zeitpunkt aus dem Ergebnisordner des zentralen Hosts ausgelesen.
- `/api/updated_tests_sorted_by_date`
  Das Resultat hat die gleiche Struktur wie Antworten auf `/api/tests_sorted_by_date`. Allerdings werden die Informationen zu den Experimenten neu ausgelesen und für spätere `/api/tests_sorted_by_date` Anfragen auf dem Server gespeichert. Die Antwort auf diesen Request kann mehrere Sekunden brauchen, da alle Experimentordner – insgesamt sind es mehrere tausend – im Ergebnisordner analysiert werden müssen.

Die statischen Dateien beinhalten zum Großteil den Quellcode für das Frontend. Jediglich die Datei `/connections.json` wird vom Frontend flexibel ähnlich zu den oben genannten Requests nachgeladen, da sie die aktuelle Netzwerktopologie des Testbeds beinhaltet.
Für die Umsetzung des Servers hätten auch andere Programmiersprachen wie Java verwendet werden können, Python bot sich aber wegen der einfachen Programmierung an.

### 4.2.2 *Webfrontend*

Mithilfe der oben genannten Informationen kann das Webfrontend visualisierte Ausgaben produzieren. Da AngularJS


doi: 10.2313/NET-2016-09-1_03

eine gute Strukturierung des Frontends erlaubt und über viele unterstützte Bibliotheken verfügt, wurde es im Frontend zusammen mit HTML verwendet. Zur Visualisierung werden die Bibliotheken vis.js und Chart.js benutzt.
Die Statusseite gliedert sich in drei Tabs. Im ersten Tab ist ein Verlaufdiagramm enthalten, dass die Anzahl der Experimente visualisiert. Dabei kann nach dem Benutzer, der die Experimente ausgeführt hat, und/oder nach den Hosts, die an den Experimenten beteiligt waren, gefiltert werden. Außerdem kann der Betrachtungszeitraum angepasst werden. Diese Seite gibt also einen guten Überblick über die Aktivität auf dem Testbed in der Vergangenheit, kann aber genauso hergenommen werden, um detaillierte Informationen zu erhalten – ähnlich zum Gantt Chart des Iot Lab. Abbildung 3 zeigt das erste Tab, welches zugleich die Startseite ist.
Im zweiten Tab können im Gegensatz dazu Live Aktivitäten betrachtet werden. Das beinhaltet aktuell eingeloggte Nutzer und den Status der einzelnen Hosts.
Das letzte Tab zeigt verschiedene Diagramme, die Daten wie die Anzahl der Hosts pro Test visualiseren und dabei keinen Verlauf berücksichtigen. Die Topologie des Netzwerks ist auch auf diesem Tab enthalten.
Da das Aktualisieren der Daten sehr lange (mehrere Sekunden) dauern kann, werden Aktualisierungen der Daten nur durch explizite Anfragen des Users durchgeführt. Die expliziten Anfragen werden durch Buttons ausgelöst, wie sie in Abbildung 3 sichtbar sind. Die Live-Daten werden zusätzlich beim Laden der Website geupdatet.

# 5. AUSBLICK

Wie der erstellten Statistikseite bzw. der Abbildung 3 entnommen werden kann, wurden bereits mehrere tausend Experimente im Baltikum Testbed erfolgreich durchgeführt. Jedoch ist die Entwicklung des Testbeds trotz der hohen Experimentezahl lange nicht beendet. Zum Beispiel wird das Erstellen und Konfigurieren von Experimenten durch eine REST-API erleichtert werden [12]. Zusätzlich sind der Seite des Lehrstuhls Netzarchitekturen und Netzdienste [6] einige Ausschreibungen für Studentenarbeiten zu entnehmen, die sich mit dem Testbed beschäftigen.

# 6. REFERENCES


[1] Ausschreibung Studentenarbeit: Comparison of Windows and Linux Networking Performance. `http://www.net.in.tum.de/fileadmin/bibtex/publications/theses/2015_9_16-Windows2.pdf`, April 2016.

[2] Ausschreibung Studentenarbeit: Smart Meter. `http://www.net.in.tum.de/fileadmin/bibtex/publications/theses/2015_06_12-MEMPHIS_SmartPDU.pdf`, April 2016.

[3] Baltikum Testbed Wiki. `https://projects.net.in.tum.de/projects/baltikum/wiki`, März 2016.

[4] Emulab Node Usage Statistics. `https://www.emulab.net/node_usage/`, März 2016.

[5] FIT IoT Lab. `https://www.iot-lab.info/`, März 2016.

[6] MEMPHIS / Baltikum Testbed. `http://www.net.in.tum.de/de/projekte/dfg-memphis/`, April 2016.

[7] PlanetLab Europe Statistics. `http://stats.planet-lab.eu/?page=1`, März 2016.

[8] P. Emmerich, S. Gallenmueller, D. Raumer, F. Wohlfahrt, and G. Carle. MoonGen: A Scriptable High-Speed Packet Generator. In *Internet Measurement Conference*, 2015.

[9] P. Emmerich, D. Raumer, F. Wohlfahrt, and G. Carle. Performance Characteristics of Virtual Switching. In *International Conference on Cloud Networking*, 2014.

[10] P. Emmerich, D. Raumer, F. Wohlfahrt, and G. Carle. Assessing Soft- and Hardware Bottlenecks in PC-based Packet Forwarding Systems. In *Fourteenth International Conference on Networks*, 2015.

[11] Stefan Kreuzer. Automation of Virtual Machine Setups for Network Experiments. Bachelor Arbeit, Technische Universität München, 2015.

[12] Tobias Betz. Optimierung des Ablaufs von testbedbasierten Netzwerkexperimenten. Bachelor Arbeit, Technische Universität München, 2015.