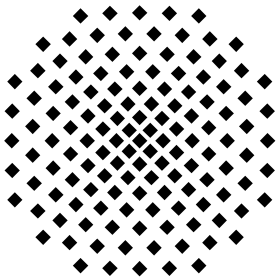



# Universität Stuttgart

## Fakultät Informatik, Elektrotechnik und Informationstechnik


Diplomarbeit Nr. 2443

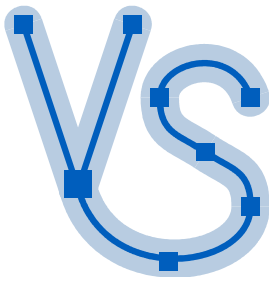
# Parametrisierbare Experimentläufe in einem Emulationssystem für Rechnernetze


Mirko Casper


| | |
|---|---|
| Studiengang: | Informatik |
| Prüfer: | Prof. Dr. K. Rothermel |
| Betreuer: | Dipl.-Inf. Steffen Maier |
| Beginn am: | 15.01.2006 |
| Beendet am: | 17.07.2006 |
| CR-Nummer: | C.2, D.3.4, I.6.2, I.6.7 |



Institut für Parallele und
Verteilte Systeme (IPVS)
Abteilung Verteilte Systeme
Universitätsstr. 38
70569 Stuttgart

# Zusammenfassung

Bei der Entwicklung verteilter Anwendungen und neuer Netzwerkprotokolle sind oftmals bereits während der Entwicklungsphase Evaluationen nötig, um deren späteren Einsatz sicherstellen zu können. Eine Möglichkeit dazu ist die Rechnernetzemulation, die bestimmte Netzwerkeigenschaften nachbildet. Für eine Untersuchung werden dazu Szenarien definiert und Experimente durchgeführt, die eine Leistungsbewertung der Testsubjekte erlauben. Im Allgemeinen ist ein Entwickler daran interessiert, bestimmte Parameter eines Szenarios zu variieren, um das Verhalten seines Testsubjekts auf veränderte Rahmenbedingungen untersuchen zu können. Es sind daher meist viele Experimente für eine umfassende Bewertung nötig. Eine integrierte Arbeitsweise durch Definition eines parametrisierten Szenarios und dessen Durchführung als Experimentlauf ist dabei wünschenswert, um auf effiziente Weise Ergebnisse erhalten zu können.

In dieser Arbeit wird eine Unterstützung für parametrisierbare Experimentläufe realisiert. Als physische Umgebung wird das Network Emulation Testbed der Universität Stuttgart eingesetzt, das ein Emulationssystem für Rechnernetze auf Basis eines PC-Clusters bereitstellt. Als Grundlage der Unterstützung dient das Software-System „Emulab“. Es verfügt bereits über Dienste zur Verwaltung von Experimenten und physischen Ressourcen. Allerdings werden in Emulab nur einzelne, unabhängige Experimente unterstützt.

Emulab wird daher erweitert, so dass eine Durchführung von parametrisierten Experimentläufen im Network Emulation Testbed möglich ist. Dazu wird die Definition von Parametern und deren Wertemengen für ein Szenario ermöglicht. Die Durchführung und Verwaltung von parametrisierten Experimentläufen wird dann durch die Anpassung und Erweiterung der bereits in Emulab bestehenden Funktionalität realisiert. Für eine automatische Durchführung von Experimentläufen wird zusätzlich eine Ablaufsteuerung bereitgestellt. Durch den Einsatz verschiedener Methoden wird dabei die Durchführungsdauer von Experimentläufen reduziert, um eine effiziente Unterstützung zu gewährleisten.

Eine Evaluation mit verschiedenen Szenarien zeigt, dass die Methoden zur Verkürzung der Durchführungsdauer erfolgreich sind. Gegenüber einer bereits in Emulab bestehenden Ablaufsteuerung werden dabei Laufzeiten gemessen, die teilweise nur die Hälfte betragen. Die Evaluation zeigt weiterhin, dass zwar eine kurzzeitig höhere Belastung der physischen Ressourcen auftritt, deren Nutzung jedoch effizienter ist.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Quelltextverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Motivation

Gegenstand derzeitiger und sicherlich auch zukünftiger Forschungsarbeiten ist die Entwicklung verteilter Anwendungen sowie die Entwicklung von Netzwerkprotokollen. Beide Bereiche haben gemeinsam, dass mehrere Teilnehmer über ein Rechnernetz kommunizieren. Das Verhalten einer neuen Anwendung oder eines neuen Netzwerkprotokolls ist durch das verteilte Zusammenspiel mehrerer Teilnehmer auf unterschiedlichen Rechnern vorab nicht immer eindeutig bestimmbar. Gerade bei zeitlichen Zusammenhängen, wie Wartezyklen bei Netzwerkprotokollen, sind oftmals auch Randbedingungen, die z.B. durch den Einsatz unterschiedlicher Betriebssysteme entstehen können, zu beachten.

So kann es zum Beispiel sein, dass in einem zu entwickelnden Netzwerkprotokoll der Zustand eines Empfängers nicht vom Sender erkannt werden kann, und daher nach einer fehlerhaften Übermittlung von Daten vor einem erneuten Senden zunächst gewartet werden muss. Die Wartezeit beeinflusst dabei allerdings meist auch die Effizienz der Kommunikation. Um eine möglichst optimale Wartezeit vor der erneuten Übertragung zu finden, kann es erforderlich sein, schon während der Entwicklung des Netzwerkprotokolls Evaluationen vornehmen zu müssen, um dessen spätere Einsatzfähigkeit zu gewährleisten. Dazu werden Experimente durchgeführt, bei denen Messungen eine Analyse des Verhaltens verdeutlichen. Emulationen sind dabei eine Möglichkeit, solche Experimente z.B. schon vor der Verfügbarkeit der späteren Zielumgebung durchführen zu können.

Im Rahmen des NET-Projekts (Network Emulation Testbed) wurde an der Abteilung Verteilte Systeme ein Rechernetzemulationssystem auf Basis eines PC-Clusters aufgebaut. Das System besteht aus einer Kombination von flexibel konfigurierbarer Hardware und Emulationswerkzeug-Software, die spezielle Netzwerkeigenschaften nachbildet (emuliert). Es ermöglicht vergleichende Leistungsmessungen von Netzwerkprotokollen und verteilten Anwendungen.

Im Allgemeinen ist ein Benutzer eines solchen Systems daran interessiert, bestimmte Parameter eines Szenarios zu variieren, um das Verhalten seines Testsubjekts auf veränderte Rahmenbedingungen zu untersuchen. Für eine eingehende Analyse sind oft viele Experimente nötig, um das Verhalten einer verteilten Anwendung oder eines neuen Netzwerkprotokolls erkennen und verstehen zu können. Im angeführten Beispiel, bei der Suche einer geeigneten Wartezeit vor einem erneuten Übertragungsversuch nach einem Fehler, sind für die Ermitt-

lung sicherlich viele Experimente unter verschiedenen Rahmenbedingungen durchzuführen. Ein vollständiges Bild ergibt sich erst, wenn z.B. die Anzahl der Teilnehmer oder Verlustraten von Verbindungen variiert werden, und damit eine eingehende Analyse des Verhaltens ermöglicht wird.

Der Einsatz von parametrisierbaren Experimentläufen ist daher eine Möglichkeit, Benutzer bei der Bewältigung solcher Aufgaben zu unterstützen und eine effiziente Arbeitweise zu ermöglichen. Eine Parametrisierung hilft dabei nicht nur Rahmenbedingungen in einem vorab bekannten Kontext zu variieren, sondern kann auch eine explorative Arbeitsweise, z.B. durch Evaluation von „was-wäre-wenn“-Szenarien, ermöglichen.

## 1.2 Aufgabenstellung

In dieser Arbeit sollen Konzeption, Architektur und ein Prototyp zur Unterstützung von Benutzern bei der Erstellung und Ausführung parametrisierter Emulationsläufe entwickelt werden. Hierzu ist zunächst eine bestehende Szenariobeschreibungssprache geeignet zu erweitern, so dass ein Benutzer variable Parameter und deren jeweilige Wertemenge, über die iteriert werden soll, spezifizieren kann. Anschließend ist eine Architektur zur Erweiterung einer bestehenden Emulationsablaufsteuerung zu entwickeln, so dass eine effiziente Hintereinanderausführung von Szenarioausprägungen unterstützt wird. Anhand einer prototypischen Implementierung ist die Effizienz des Ansatzes zu zeigen.

Für nicht parametrisierte Emulationsläufe existiert bereits die Software „Emulab“. Entwickelt wurde diese im Rahmen des „Emulab Network Testbed Projekts“ an der University of Utah. Emulab übernimmt u.a. die vollständige Steuerung eines Emulationssystems und bietet hierfür beispielsweise Dienste zur Benutzerverwaltung, Zugangskontrolle und Ressourcenverwaltung. Für einzelne Experimente wird die Ablaufsteuerung bestehend aus Ressourcenanforderung, Konfiguration, Ereignissteuerung des eigentlichen Ablaufs, Ergebnissammlung und Ressourcenfreigabe übernommen. In dieser Arbeit ist diese Software entsprechend zu erweitern. Dies beinhaltet die Inbetriebnahme eines hierfür notwendigen Kernsystems der Emulab Steuerungs-Software im Network Emulation Testbed.

## 1.3 Aufbau der Arbeit

Die Arbeit gliedert sich insgesamt in acht Kapitel. In Kapitel 2 werden zunächst das Network Emulation Testbed (NET) und Emulab vorgestellt, und der Einsatz des Emulab-Systems in NET diskutiert. Nach einer Erörterung verwandter Arbeiten in Kapitel 3, wird in Kapitel 4 näher auf die Aspekte von Szenariobeschreibungen eingegangen. Danach wird die Szenariobeschreibungssprache von Emulab erläutert und der Lebenszyklus und die Durchführung von Experimenten in Emulab genauer betrachtet.

Die Erkenntnisse werden in Kapitel 5 herangezogen, um einen zweckmäßigen Ansatz für eine integrierte Unterstützung parametrisierter Experimentläufe in Emulab zu entwickeln. Um eine effiziente Unterstützung bei der Durchführung parametrisierter Experimentläufe zu gewährleisten, werden in Kapitel 6 Konzepte für eine Ablaufsteuerung diskutiert und deren Umsetzung im Kontext des Emulab-Systems erörtert.

Die prototypische Implementierung wird anschließend in Kapitel 7 evaluiert. Mit Hilfe verschiedener Szenarien werden parametrisierte Emulationsläufe durchgeführt und mit der bereits bestehenden Ablaufsteuerung verglichen, um die Vorteile des gewählten Konzepts darzustellen. Eine Zusammenfassung der Arbeit und Hinweise auf mögliche Erweiterungen werden abschließend in Kapitel 8 gegeben.

# Kapitel 2

# Grundlagen

Im vorigen Kapitel wurden bereits die beiden Systeme genannt, die für eine Unterstützung von parametrisierbaren Experimentläufen in dieser Arbeit verwendet werden: Das Network Emulation Testbed und Emulab. Daher werden in diesem Kapitel zunächst beide Systeme vorgestellt und deren Software und Hardware beschrieben, soweit sie für das Verständnis in den weiteren Kapiteln benötigt werden.

## 2.1 Network Emulation Testbed

Das Network Emulation Testbed (NET) der Abteilung Verteilte Systeme an der Universität Stuttgart ist ein flexibel vernetzter Linux PC-Cluster, in dem beliebige Netztopologien und deren Eigenschaften nachgebildet werden können. Damit bietet das NET eine ideale Analyseumgebung für verteilte Anwendungen und Netzprotokolle [HR02].

In diesem Abschnitt wird ein Überblick über die eingesetzte Hard- und Software und die derzeitige Verwendung des Network Emulation Testbed gegeben.

### 2.1.1 Hardware

Der NET-Cluster besteht aus 64 gleichen Knoten, die über zwei Ethernet Netzwerke und zwei Vermittlungsstellen miteinander gekoppelt sind. Das erste Netz dient zu Verwaltungs- und Administrationsaufgaben und besitzt eine Bandbreite von 100 Mbps.

Das zweite Netz stellt das eigentliche Experimentiernetz dar. Das Experimentiernetz wird über einen Switch gekoppelt, der alle Knoten mit einer Bandbreite von 1 Gbps vollduplex verbindet (siehe Abb. 2.1). Der Zugriff auf den Cluster wird über zwei Frontend-Rechner ermöglicht, die auch zur zentralen Datenhaltung und zu Administrationsaufgaben dienen.

Jeder Knoten des Clusters besitzt die gleiche Ausstattung. Ausgerüstet ist er mit einem 2,4GHz Intel Pentium 4 Prozessor, einem Hauptspeicher von 512MB und einem Frontside-Bus mit 133MHz Taktrate. Das Administrationsnetz wird über eine Intel EtherPro Netzwerkkarte angesprochen. Die Anbindung an das Experimentiernetz findet über eine Realtek Semiconductor Gigabit-Ethernetkarte (RTL-8169) statt, die über einen 33MHz, 32bit PCI-Bus an den Rechner angeschlossen ist.

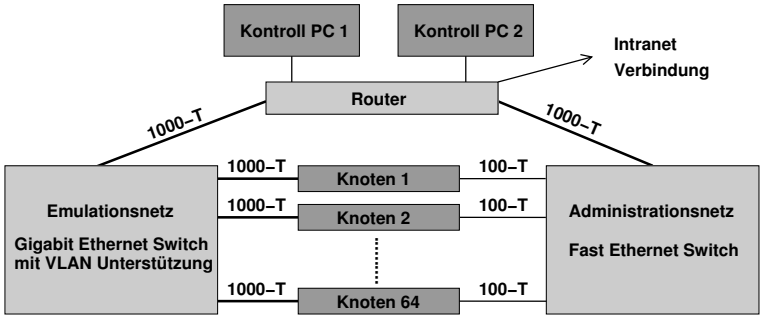



Abbildung 2.1: Hardware-Architektur von NET

Alle Knoten sind gekoppelt über einen FastIron II+ Switch von Foundry Networks [Fou04, Fou03]. Der Switch ist modular in einer zweistufigen Architektur aufgebaut. Eine Backplane dient zur Aufnahme von bis zu acht Steckplätzen, die über ein voll vermaschtes, internes Netz miteinander verbunden sind. Die Steckplätze können verschiedene Interface Module aufnehmen. Im NET werden 1-Gbps-Module verwendet, die über acht Ports verfügen und als Portgruppe bezeichnet werden. Der Einsatz von acht Portgruppen mit jeweils acht Ports verbindet dann die 64 Knoten des Clusters.

Die Umsetzung einer Netzwerktopologie für ein Emulationsszenario wird mit Hilfe des Switch erreicht [HM04]. Verschiedene VLANs (Virtual LANs) bilden dabei die Kommunikationskanäle der beteiligten Knoten. Ein VLAN mit nur zwei Knoten stellt somit eine exklusive Verbindung zwischen den Knoten dar. Die Grundlage für einen Kommunikationskanal mit gemeinsamem Medium wird durch ein VLAN gebildet, das mehrere Knoten umfasst. Jeder physische Knoten entspricht dabei genau einem emulierten Knoten.

### 2.1.2 Software

Alle Knoten im Network Emulation Testbed besitzen die gleiche Softwareausstattung. Als Betriebssystem wird Linux eingesetzt, die derzeit verwendete Distribution ist RedHat 7.3. Um Emulationen durchführen zu können, ist der Linux-Kernel der Knoten um verschiedene Module und Patches erweitert worden, und es werden weitere Emulationswerkzeuge verwendet, die im Folgenden näher beschrieben werden.

#### Emulationswerkzeuge

Der NETShaper dient zur Nachbildung von Eigenschaften eines Kommunikationskanals. Als mögliche Parameter können die Bandbreite des Kommunikationskanals limitiert, feste und variable Verzögerungen bestimmt und Rahmenverluste emuliert werden.

Die Implementierung beinhaltet zwei Programme. Ein Kernel-Modul und einen Konfigurations-Dämon. Der NETShaper ist im Kernel als Netzwerkschnittstelle implementiert, die als Emulationsschicht wie in Abb. 2.2 dargestellt im Protokollstapel verankert wird.

Zur Kommunikation über ein Netzwerk wird eine NETShaper-Instanz an eine real existierende Schnittstelle gebunden. Von höheren Schichten wird sie ebenfalls wie eine reale Netzwerkschnittstelle angesprochen und ist somit transparent eingebunden. Die nachzubildenden

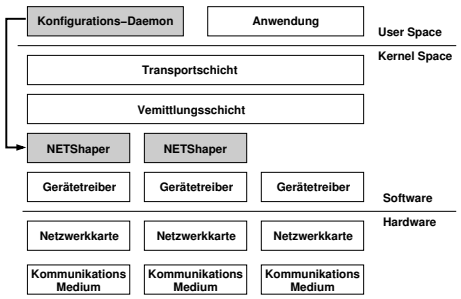



Abbildung 2.2: Einordnung des NETShaper im Protokollstack

Eigenschaften können während der Initialisierung des Kernel-Moduls oder mittels des Konfigurations-Dämon dynamisch zur Laufzeit geändert werden.

Bei der Emulation großer Szenarien ist die Zahl der beteiligten Knoten oft größer als die im NET zur Verfügung stehenden 64 Knoten. Die Skalierbarkeit der emulierbaren Szenarien wird erweitert, indem eine Ressourcenvirtualisierung vorgenommen wird. Dazu werden mehrere Werkzeuge eingesetzt. Eine Komponente dient zur Verwaltung virtueller Sockets und Routing-Tabellen [Leu04]. Der VNMux implementiert als Teil der Ressourcenvirtualisierung die Funktionalität eines virtuellen Switches auf einem Knoten, und ist ebenfalls als Kernel-Modul implementiert. Auf einem physischen Knoten (pnode) kann damit die Emulation mehrerer virtueller Knoten (vnodes) erfolgen, die über ein virtuelles Netz kommunizieren. Mehrere vnmux-Instanzen ermöglichen es, auf einem Knoten mehrere Netze zu emulieren. Die Konfiguration erfolgt analog zu NETShaper auf zwei Arten. Beim Laden des Kernel-Moduls können Optionen angegeben werden, die zur Initialisierung dienen, dynamische Änderungen ermöglicht das Programm „vnmuxcfg“.

### Werkzeuge zur Durchführung und Steuerung von Experimenten

Neben Werkzeugen zur Nachbildung von Verbindungseigenschaften werden bei der Durchführung von Experimenten Programme eingesetzt, die den Ablauf steuern.

Der Mobile Scenario Controller (MSC) steuert den Ablauf eines Experiments mit dynamischer Topologie, die aufgrund der Mobilität der Kommunikationsteilnehmer entstehen kann. Die gesamte Steuerung wird dazu zentral vom Mobile Scenario Controller koordiniert. Die Mobilität wird dann durch dynamische Änderungen der Verbindungseigenschaften emuliert. Die Änderungen werden dazu vom Mobile Scenario Controller an die verteilten Emulationswerkzeuge der beteiligten Knoten über ein eigenes Protokoll übermittelt.

### Administrative Werkzeuge

Weitere Werkzeuge umfassen meist administrative Aufgaben, wie zum Beispiel die Verwaltung von Benutzerzugängen oder die Installation und Wartung von Knoten.

### 2.1.3 Einsatzgebiete und Beispielszenarien

Das Network Emulation Testbed der Universität Stuttgart wird hauptsächlich für Evaluationen von Protokollen und verteilten Anwendungen in zwei Bereichen eingesetzt. Den größten Anteil bilden Emulationen von mobilen ad-hoc Szenarien. Die Evaluation von Protokollen und verteilten Anwendungen in Internet-Umgebungen stellen das zweite Einsatzgebiet dar. Für beide Einsatzgebiete werden im Folgenden Szenarien vorgestellt, die in den weiteren Kapiteln als Beispiele unterstützend herangezogen werden.

#### Beispiel 1: MANET Szenario

Ein mobiles ad-hoc Netz (MANET) verbindet mobile Computer. Es wird gebildet durch eine Anzahl gleichberechtigter, mobiler Kommunikationspartner, die miteinander kommunizieren, ohne auf weitere Infrastruktur oder Koordinierungsstellen zurückgreifen zu müssen. Eine Anbindung an ein fest installiertes Netz ist dabei nicht nötig. Die Signalübermittlung erfolgt in einem MANET drahtlos, meist wird Funkübertragung eingesetzt.

Dadurch unterscheidet es sich in seinen Eigenschaften deutlich von traditionellen, kabelgebundenen Rechnernetzen. Der Kommunikationskanal ist durch drahtlose Signalübermittlung unzuverlässig, und es werden oft nur geringe Übertragungsraten erreicht. Weiterhin ist die Anzahl der Kommunikationspartner meist groß, deren Leistung (Rechenkapazität, Speicher) aber gering (vgl. [Rot02],[Sch00]). Durch die Mobilität der Teilnehmer kann sich die Topologie des Netzes kontinuierlich ändern. Da die Reichweite für Übertragungen begrenzt ist, werden für die Kommunikation über größere Entfernungen Verfahren benötigt, die eine Weiterleitung von Daten über Zwischenstationen erlauben. Als Folge zu großer Entfernungen der Teilnehmer voneinander, können auch Partitionierungen des Netzwerks auftreten.

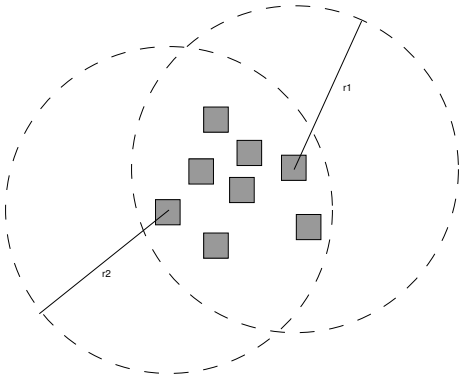



Abbildung 2.3: Topologie des MANET Szenarios

Die Entwicklung neuer Routing-Protokolle für solche Netzumgebungen ist eine komplexe Aufgabe. Aufgrund sich stetig ändernder Verbindungseigenschaften, können Probleme entstehen, die vorab oft nur unvollständig erkannt werden können. Die Evaluation von Routing-Protokollen stellt damit einen wichtigen Aspekt im Entwicklungszyklus dar, um vor deren eigentlichem Einsatz die Leistungsfähigkeit und Grenzen in verschiedenen Situationen zu erkennen.

In Abb. 2.3 ist ein einfaches Szenario für die Evaluation eines neuen Routing-Protokolls dargestellt. Das Szenario umfasst acht Kommunikationsteilnehmer, die durch acht Knoten repräsentiert werden. Es wird für das Szenario angenommen, dass sich anfänglich alle Teilnehmer in gegenseitiger Kommunikationsreichweite befinden. Während des Experiments verändern die Teilnehmer (z.B. nach dem Random Waypoint Mobilitäts-Modell) ihren Standort. Dementsprechend ändern sich ebenfalls die Eigenschaften der Verbindungen zwischen den Knoten (z.B. Rahmenverluste). Durch die Mobilität kann eine direkte Kommunikation der Teilnehmer bei einer zu großen Entfernung voneinander nicht mehr möglich sein, so dass andere Teilnehmer bzw. Knoten vermitteln müssen.

**Beispiel 2: Internet Szenario**

Ein weiteres Anwendungsgebiet des Network Emulation Testbed ist die Evaluation in leitungsgebundenen Umgebungen, wie zum Beispiel dem Internet. Ein Ziel von Experimenten in dieser Umgebung könnte es sein, das Verhalten einer verteilten Anwendung unter verschiedenen Randbedingungen zu evaluieren. Die Verbindungsparameter der einzelnen Kommunikationskanäle sind während eines solchen Experiments meist unverändert, oft werden fehlerfreie Kommunikationskanäle angenommen. Dennoch können Überlastsituationen im Netz zu unerwünschtem Verhalten einer verteilten Anwendung führen.

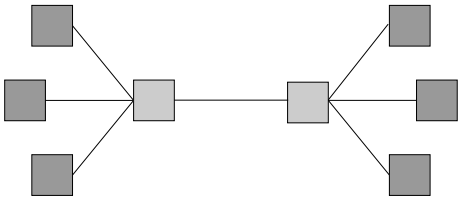


Abbildung 2.4: Topologie des Internet Szenarios

Abb. 2.4 zeigt eine einfache Topologie, die zur Evaluation des Verhaltens einer verteilten Anwendung bei Überlast einer Teilstrecke herangezogen werden kann. Die Topologie stellt ein einfaches „dumbbell“-Schema dar, das mehrere Knoten über zwei Router miteinander verbindet. Bei steigender Last stellt die Verbindung zwischen beiden Routern dann einen möglichen Flaschenhals dar. Die Auswirkungen einer Überlast im Netz auf eine verteilte Anwendung können durch Evaluationen in diesem Szenario erkannt werden.

## 2.2 Emulab

Emulab ist ein System, das zur Durchführung von Experimenten im Bereich von verteilten Systemen und Netzen in Forschung, Entwicklung und Lehre eingesetzt werden kann [WLS+02]. Entwickelt wurde das System an der University of Utah, USA.

Experimente können in Emulab im Kontext von Simulation, Emulation oder in realem Umfeld erfolgen. Emulab integriert den Einsatz dieser Ausführungsumgebungen in einem einzigen System. Dazu stellt es Abstraktionen, Dienste und Namensräume für alle Umgebungen zur Verfügung. Durch die Beschreibung von Szenarien in einer generischen Form und deren

automatischen Umsetzung in verschiedene Ausführungsumgebungen, werden die heterogenen Details der verschiedenen Ansätze verborgen. Daher können auf einfache Weise übergreifend vergleichende Messungen oder Validierung von Messergebnissen in den verschiedenen Umgebungen vorgenommen werden, oder für ein Szenario die „geeignetste“ Umgebung gewählt werden.

Im Folgenden wird zunächst die Hardware-Architektur von Emulab und danach das Software-System näher betrachtet, und ein Überblick über deren Komponenten gegeben.

### 2.2.1 Hardware

Zur Durchführung von Experimenten werden von Emulab physische Ressourcen verwaltet und bereitgestellt. Abb. 2.5 stellt die grundsätzliche Hardware-Architektur dar, die für einen Einsatz des Emulab-Systems bereitgestellt werden muss. Die Architektur ist physisch in mehrere Komponenten gegliedert: drei Frontend-Rechner, ein Verbindungsnetzwerk und mehrere Cluster-Knoten.

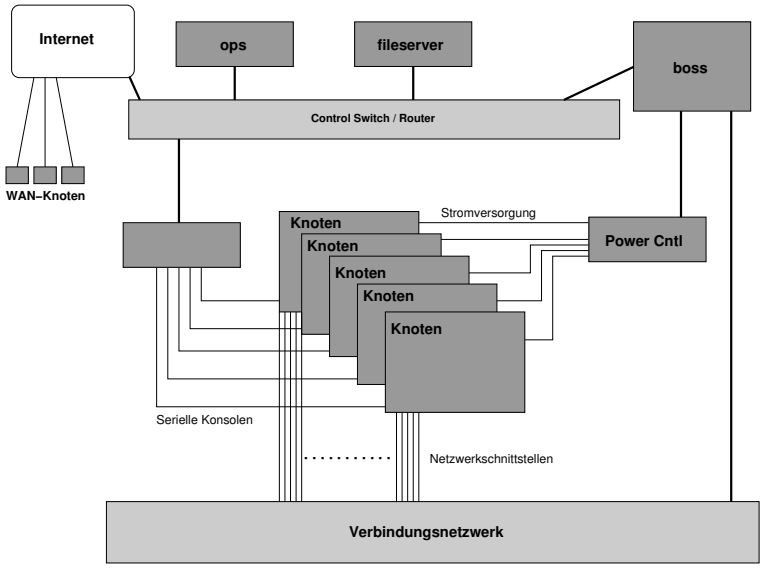



Abbildung 2.5: Hardware-Architektur von Emulab

Die drei Frontent-Rechner dienen zur Verwaltung und Steuerung des Systems. Ein Steuerungsrechner (boss) übernimmt die Aufgabe der Ressourcenverwaltung und der Steuerung von Experimenten, sowie der Speicherung der dazu notwendigen Daten. Der Zugriff auf die Cluster-Knoten ist für Benutzer durch den zweiten Frontend-Rechner (ops) möglich. Der Einsatz eines dedizierten Rechners zur Speicherung von Benutzerdaten (fileserver) ist optional. In Emulab ist es möglich, diese Daten auch auf „ops“ zu speichern.

Die Frontend-Rechner und Cluster-Knoten sind über ein Verbindungsnetzwerk miteinander gekoppelt. Die Architektur des Verbindungsnetzwerkes kann dabei nahezu beliebig sein und aus mehreren Koppelelementen bestehen. Die gesamte Verbindungstruktur der Koppelelemente kann dabei als eine Art „programmierbares Patchpanel“ betrachtet werden.

Die Cluster-Knoten werden zur Durchführung von Experimenten eingesetzt. Dazu verfügen sie über mindestens zwei Schnittstellen zum Verbindungsnetzwerk. Eine Schnittstelle dient zur Übermittlung von administrativen Daten, alle weiteren können für Experimentierzwecke verwendet werden. Zusätzlich zu lokalen Knoten können für Experimente auch geographisch verteilte Knoten über das Internet eingebunden werden (WAN-Knoten).

Zur Verwaltung der lokalen Knoten unterstützt Emulab den optionalen Einsatz von Hardware, die zur Steuerung der Stromversorgung der Knoten (power control) eingesetzt werden kann. Knoten, die z.B. bei Zugriffen über das Verbindungsnetzwerk nicht mehr ansprechbar sind, können damit im Fehlerfall durch eine kurzzeitige Unterbrechung der Stromversorgung wieder zurückgesetzt werden. Durch den Anschluss von seriellen Konsolen besteht zudem die Möglichkeit, Knoten auf einem alternativen Weg schon vor dem Laden eines Betriebsystems zu steuern. Ein Zugriff auf die seriellen Konsolen ist für Benutzer über den Frontend-Rechner „ops“ möglich.

### 2.2.2 Software

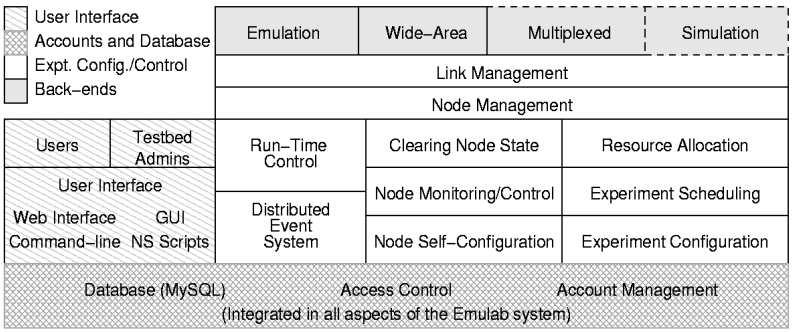



Abbildung 2.6: Software-Architektur von Emulab [Gro05f]

Die Emulab-Software ist, wie in Abb. 2.6 dargestellt, in mehrere Komponenten gegliedert. Alle Daten zur Benutzer- und Experimentverwaltung werden zentral in einem Datenbank-Backend gespeichert und von allen Komponenten des Systems verwendet. Dort gespeicherte Benutzerdaten werden z.B. zur Authentifikation für den Zugriff auf das System verwendet. Benutzern wird zur Interaktion mit dem System eine web-basierte Oberfläche bereitgestellt. Alternativ besteht die Möglichkeit, das System über Aufrufe von Kommandozeilen-Programmen zu steuern.

Die Verwaltung und Steuerung von Experimenten stellen die Kernfunktionalitäten von Emulab dar. Emulab bietet Möglichkeiten, Experimente in verschiedenen Umgebungen durchzuführen. Neben Emulationen sind auch Simulationen und Experimente in realen Internet-Umgebungen möglich. Da Experimente auf generische Weise unabhängig von der Zielumgebung verwaltet werden, können diese Komponenten als nahezu unabhängige Backend-Module betrachtet werden.

Die zur Durchführung von Experimenten notwendige Funktionalität ist in mehreren Komponenten realisiert. Dazu gehören z.B. Möglichkeiten zur Reservierung von Ressourcen und

Konfiguration von Knoten und des Verbindungsnetzwerks. Während der Ausführung stellen Schnittstellen zur Steuerung des Ablaufs eine Möglichkeit dar, dynamische Aspekte eines Szenarios zu kontrollieren. Als Betriebssystem wird auf den Frontend-Rechnern FreeBSD eingesetzt.

### Benutzerverwaltung

Benutzer können auf zwei Arten auf Emulab zugreifen. Alle Aufgaben können über eine webbasierte Oberfläche durchgeführt werden, oder durch Ausführen von Kommandozeilen-Programmen auf „ops“ erfolgen.

Dazu müssen Benutzer Zugriff auf das System haben. Ein Berechtigungssystem steuert die Rechte, die ein Benutzer im Umgang mit dem System erhält. Das System definiert dazu verschiedene Rollen. Alle Benutzer des Systems haben die Möglichkeit Experimente auszuführen und nehmen daher die Rolle eines Experimentators ein. Anwender in der Rolle des Administrators besitzen zusätzliche Rechte, um Verwaltungsaufgaben wahrnehmen zu können.

Die Organisation der Experimente und Benutzer ist dabei hierachrisch strukturiert. Dazu werden Experimente und Benutzer Projekten bzw. Projektgruppen zugeordnet. Jedes Projekt wird dabei von einem Projektleiter koordiniert. Vor Durchführung eines Experimentes muss daher zunächst ein Projekt angelegt werden. Die Genehmigung dazu obliegt den Administratoren. Die Aufnahme weiterer Benutzer in ein Projekt wird dann an den Projektleiter delegiert. Innerhalb eines Projektes können zusätzlich mehrere Gruppen angelegt werden, die sowohl zur Gliederung von Experimenten, als auch zur Gruppierung von Benutzern innerhalb eines Projekts verwendet werden können [Gro05a].

### Experimente

Experimente stellen die Arbeitseinheit dar, mit denen ein Benutzer von Emulab umgeht. Experimente werden durch ein Szenario beschrieben. Ein Szenario stellt eine virtuelle, von der Ausführungsumgebung unabhängige Beschreibung eines Experiments dar. Sie wird in Emulab in einer ns-kompatiblen Syntax definiert (ns - Network Simulator). Die Beschreibung beinhaltet dabei sowohl statische Aspekte, wie z.B. die Menge der Knoten und deren Eigenschaften und die gewünschte Topologie, als auch dynamische Aspekte, die den Ablauf eines Experiments steuern und Änderungen der Eigenschaften zur Laufzeit erlauben. Da die Beschreibung nahezu unabhängig von der späteren Ausführungsumgebung ist, kann ein Experiment fast ohne Änderung z.B. als Emulation oder Simulation laufen. Das Anlegen eines Experimentes erfolgt über die Web-Oberfläche durch Zuordnung eines Experiments zu einem Projekt und Übergabe einer Szenariobeschreibung.

Die Durchführung eines Experiments gliedert sich dann in drei Phasen: Einer Startphase, der eigentlichen Durchführungsphase und einer Endphase. In der Startphase wird ein Szenario (die virtuelle Beschreibung) in eine physische Repräsentation umgesetzt und eine Ausführungsumgebung von Emulab automatisch konfiguriert.

Danach beginnt die Durchführungphase eines Experiments. Zur Unterstützung werden einem Experimentator dazu auf den Knoten verschiedene Werkzeuge bereitgestellt. Zusätzlich

verfügt Emulab über ein Ereignissystem, das dynamische Änderungen eines Szenarios zur Laufzeit erlaubt.

In der Endphase werden die vorgenommenen Konfigurationsänderungen zur Bereitstellung der Ausführungsumgebung von Emulab wieder rückgängig gemacht. Dabei werden die reservierten Ressourcen freigegeben und deren Konfiguration zurückgesetzt, so dass sie als freie Ressource für weitere Experimente wieder zur Verfügung stehen. Eine detaillierte Beschreibung des Lebenszyklus und der Durchführung von Experimenten wird in Kapitel 4 gegeben.

### Ablauforganisation

Einem Benutzer werden zur Durchführung von Experimenten zwei Möglichkeiten gegeben, die in Emulab durch zwei Betriebsarten realisiert sind.

Der interaktive Betrieb ist eine Möglichkeit, Experimente durchzuführen, die eine direkte Einflussnahme des Benutzers voraussetzen. Durch Kommandos an das System bestimmt ein Benutzer dazu Start- und Endzeitpunkte eines Experiments manuell.

Ein einfacher Stapelbetrieb ermöglicht die Durchführung von Experimenten, die keine direkte Interaktion eines Benutzers während der Durchführungsphase erfordern. Ein Experiment wird dabei automatisch durchgeführt, indem in der Szenariobeschreibung ein Startkommando definiert wird. Das Startkommando wird dann nach der Startphase ausgeführt, und übernimmt die Steuerung der Durchführungsphase. Ein Experiment wird zur Ausführung daher lediglich einer Warteschlange übergeben, und der tatsächliche Startzeitpunkt von Emulab gesteuert. Nach Beendigung des Startkommandos wird ein solches Experiment dann automatisch beendet und die Endphase eingeleitet

Für beide Betriebsarten wird der Ablauf nicht geplant. Experimente werden aktiviert, wenn genügend physische Ressourcen reserviert werden können. Schlägt eine Aktivierung wegen Ressourcenmangels fehl, muss der Versuch bei interaktiven Experimenten vom Benutzer manuell wiederholt werden. Im Stapelbetrieb wird der Versuch für ein Experiment zu einem späteren Zeitpunkt automatisch wiederholt.

### Knoten- und Verbindungsverwaltung

Zur Durchführung eines Experiments werden vom System Ressourcen während dessen Dauer reserviert. Eine Ressource ist zu einem Zeitpunkt genau einem Experiment zugeordnet, wird aber durch Reservierung mehrerer zeitlich getrennter Experimente mehrfach verwendet.

Deshalb muss vor dem Ausführen eines Experiments sichergestellt werden, dass alle reservierten Knoten einen definierten Ausgangszustand haben, der nicht durch ein vorheriges Experiment beeinflusst wird. Das beinhaltet ebenfalls die Installation eines Betriebssystems, das in Emulab durch Aufspielen von Festplattenabbildern realisiert wird. Dafür wird ein eigens entwickeltes System, genannt Frisbee, eingesetzt [HSL+03]. In Emulab werden dazu Abbilder von Partitionen und Festplatten verwaltet, die in einer Szenariobeschreibung eines Experiments gewählt werden können. Daher können zur Durchführung von Experimenten auch verschiedene Betriebssysteme verwendet werden. Unterstützt werden derzeit FreeBSD, Linux und Windows in verschiedenen Versionen.

Neben der direkten Verwendung physischer Knoten bei einer Emulation ist auch der Einsatz von virtualisierten Ressourcen möglich. Simulatorbasierte Emulationen sind dabei eine Art der Virtualisierung und werden mittels ns-e [Fal99] realisiert. Eine weitere Möglichkeit der Virtualisierung stellt in Emulab der Einsatz von BSD-Jails und einer Komponente für virtualisiertes Routing unter FreeBSD dar [GSHL03].

Die Topologie eines Szenarios wird durch VLANs (virtual LANs) bereitgestellt. Eigenschaften der Verbindungen eines Szenarios, wie z.B. Rahmenverluste oder Bandbreiten, werden auf zwei Arten nachgebildet. Zum einen werden Emulationswerkzeuge im Betriebssystem auf den Knoten eingesetzt [Riz97, Wer98]. Zum anderen, wenn Änderungen des Betriebssystems nicht möglich sind, können speziell konfigurierte Knoten, so genannte *Delay-Nodes*, transparent zwischen die Verbindung von Knoten eingefügt werden. Alle Daten werden dann über den Delay-Node geleitet, der die in der Szenariobeschreibung definierten Eigenschaften einer Verbindung umsetzt.

## 2.3 Emulab im Network Emulation Testbed

Im Rahmen der Diplomarbeit wird das Emulab-System im Network Emulation Testbed der Universität Stuttgart eingesetzt und erweitert. Daher gilt es zunächst die Einsatzfähigkeit der Emulab-Software in NET zu untersuchen.

#### Vergleich zwischen Emulab und NET

Der Vergleich der Hardware-Architekturen beider Systeme zeigt einen hohen Deckungsgrad. Die vorhandene Architektur von NET passt sich dabei gut in die Hardware-Anforderungen von Emulab ein. Die für den Betrieb benötigten zwei Frontend-Rechner sind bereits vorhanden. Auf den Einsatz eines dedizierten Rechners zur Datenhaltung kann verzichtet werden.

Die beliebige Topologie des Verbindungsnetzwerkes zwischen den beteiligten Komponenten ist in Emulab flexibel konfigurierbar und wird in NET durch einen einzigen Switch für das Experimentiernetz realisiert. Während in Emulab nur ein Verbindungsnetzwerk existiert, welches das administrative Netz und Experimentiernetz umfasst, sind in NET beide Netze physisch getrennt. Emulab unterstützt diese Architektur des Verbindungsnetzwerkes ebenfalls.

Ein direkter Zugriff über serielle Konsolen ist in NET nur für wenige Knoten möglich. Der Zugriff auf serielle Konsolen durch Benutzer ist nicht vorgesehen und bleibt den Administratoren vorbehalten. Softwaregesteuerte Spannungsversorgung wird in NET derzeit ebenfalls nicht unterstützt. Beides sind jedoch optionale Komponenten in Emulab.

In Emulab werden von Knoten mehrere Netzwerkschnittstellen bereitgestellt, um mehrere Verbindungen zu anderen Knoten in einem Szenario nachbilden zu können. In NET besitzen alle physischen Knoten lediglich eine Schnittstelle zum Experimentiernetz. Der Einsatz von tagged VLANs kann diese Einschränkung durch Multiplexing aller Verbindungen eines Knotens aus der Szenariobeschreibung über diese eine physische Netzwerkschnittstelle aufheben. Damit Knoten aus dem Internet erreichbar sind, werden in Emulab den Knoten normalerweise offizielle IP-Adressen zugewiesen. Für Knoten in NET ist ein Zugriff über das Internet nicht vorgesehen, so dass ein privater Adressbereich verwendet werden kann.

Der Vergleich der Software-Anforderungen von Emulab mit dem derzeitigen Stand in NET zeigt einen weiteren Unterschied. Als Betriebssystem wird in Emulab für die Frontend-Rechner FreeBSD eingesetzt, in NET jedoch Linux.

### Einordnung der Diplomarbeit

Da Emulab für FreeBSD konzipiert wurde, in NET aber weiterhin Linux auf den Steuerungsrechnern zum Einsatz kommen soll, muss Emulab portiert werden. Im Rahmen der Diplomarbeit wird allerdings zunächst nur ein Kernsystem von Emulab für den Einsatz unter Linux umgesetzt. Dabei wird auf die Unterstützung des vollen Umfangs der Funktionalität zunächst verzichtet.

Einschränkungen werden dazu im Bereich der möglichen Ausführungsumgebungen gemacht. Ein Einsatz von Simulationen, simulatorbasierten Emulationen und Experimenten mit WAN-Knoten wird nicht unterstützt. Der Einsatz ist auch zu einem späteren Zeitpunkt nicht vorgesehen. Als Betriebssystem der Cluster-Knoten wird nur Linux (RedHat 9.0) eingesetzt.

Zur Durchführung von Experimenten werden bei Emulationen Verbindungseigenschaften nachgebildet. Die Nachbildung der Verbindungseigenschaften wird dabei durch verschiedene Emulationswerkzeuge auf den Knoten vollzogen. Im Rahmen der Diplomarbeit werden diese Werkzeuge zunächst aus Emulab übernommen. Die Integration der im Rahmen des NET-Projekts entwickelten Werkzeuge ist für einen späteren Zeitpunkt vorgesehen. Der Einsatz von Delay-Nodes wird daher ebenfalls nicht unterstützt.

In Emulab existieren durch simulatorbasierte Emulationen und weitere Werkzeuge Möglichkeiten zur Virtualisierung. In NET ist eine eigene Virtualisierungslösung vorhanden (siehe Abschnitt 2.1.2). Deren Integration ist ebenfalls für einen späteren Zeitpunkt vorgesehen, und die Virtualisierung von Emulab wird daher nicht für den Einsatz in NET angepasst.

Es werden also zunächst nur physische Knoten für die Durchführung von Experimenten unterstützt, auf denen durch bereits in Emulab existierende Emulationswerkzeuge Verbindungseigenschaften eines Szenarios nachgebildet werden.

# Kapitel 3

# Verwandte Arbeiten

Experimente, sei es mittels Simulation, Emulation oder in realen Umgebungen, werden in einem bestimmten Kontext durchgeführt. Die Rahmenbedingungen und Modelle, die diesen Kontext beschreiben, müssen vorab definiert werden. Die Erstellung einer Szenariobeschreibung ist also einer der ersten Schritte bei der Durchführung von Experimenten. Im Folgenden werden verschiedene Ansätze diskutiert, die diese Aufgabe verwirklichen.

Dabei lassen sich zur Beschreibung von Szenarien verschiedene Ansätze erkennen. Eine Definition von Szenarien ist durch eine Beschreibung in einer Programmiersprache, in einer Skriptsprache oder in einem strukturierterten Textformat möglich. Je nach Art wird eine Beschreibung dabei zunächst übersetzt und direkt in ein Simulations- bzw. Emulationswerkzeug eingebunden, während der Durchführung eines Experiments interpretiert oder als Datei einem Werkzeug übergeben.

Die beiden Simulationsumgebungen OMNet++ und REAL verwenden Programmiersprachen zur Beschreibung von Szenarien. OMNet++ (Objective Modular Network Testbed in C++) ist eine modular aufgebaute Simulationsumgebung. Primär wurde sie zur Evaluation von Programmen in Netzumgebungen entwickelt. OMNet++ stellt zur Definition von Szenarien die Sprache NED (NEtwork Description) bereit [VP97, Var98]. NED verwendet zur Beschreibung das Konzept von Modulen als Abstraktion. Ein Modul implementiert dabei ein bestimmtes Verhalten einer Komponente eines Szenarios und kommuniziert über Schnittstellen mit anderen Modulen. Für ein Modul kann dazu eine beliebige Anzahl von Schnittstellen definiert werden, über die eine Kommunikation mit anderen Modulen möglich wird. Module können aber auch andere Module (Submodule) beinhalten. Zur Definition einer Beschreibung werden Konstrukte wie z.B. Schleifen oder bedingte Anweisungen bereitgestellt. Eine Parametrisierung von Modulen wird ebenfalls unterstützt. Zur Durchführung von Experimenten wird eine in NED geschriebene Szenariobeschreibung durch einen Compiler in C++-Code transformiert, der dann übersetzt und direkt in den Simulator gebunden wird. Dynamische Änderungen der Module, und damit das Verhalten der simulierten Komponente, sind zur Laufzeit einer Simulation möglich. Eine benutzerfreundliche Definition wird über GNED, einen graphischen Editor, ermöglicht.

REAL (REalistic And Large) ist eine Umgebung für die Simulation von paketorientierten Netzen [Kes88, Kes89]. Entwickelt wurde sie an der Universiy of California. REAL basiert auf dem NEST-Toolkit (NEtwork Simulation Toolkit) der Columbia University [BDSY88], und er-

möglicht es, verschiedene Transportprotokolle und Überlastkontrollalgorithmen in Netzwerken zu analysieren. Szenarien werden in REAL durch die „NetLanguage“ definiert. Die Syntax der NetLanguage ist ähnlich zu Strukturen in C. Eine Beschreibung in der NetLanguage wird vor der Durchführung eines Experiments analog zu OMNet++ zunächst in C-Code transformiert. Zur Durchführung eines Experiments wird die Beschreibung dann als Modul in das Simulatorprogramm eingebunden.

Alle bisher vorgestellten Ansätze verwenden zur Definition von Beschreibungen Programmiersprachen, die über einen Zwischenschritt umgeformt und dann direkt im Werkzeug verankert werden. Ähnliche Ansätze sind auch bei der Beschreibungssprache für Parsec [BMT+98] zu finden. Bei den folgenden Simulations- bzw. Emulationsumgebungen werden zur Beschreibung von Szenarien Skriptsprachen eingesetzt.

Ns-2, ein Simulator zur Analyse von Protokollen und Netzen, wurde im Rahmen des VINT Projekts [Baj98] mit Unterstützung der DARPA als Variante des REAL-Simulators entwickelt. Ns-2 ist in zwei Sprachen implementiert. Zeitkritische Funktionen, z.B. zur Nachbildung von Netzwerkeigenschaften und Simulation von Netzwerkverkehr, werden in C++ implementiert. Szenariobeschreibungen werden in der Skriptsprache oTcl erstellt. OTcl ist eine Erweiterung von Tcl für objektorientierte Programmierung [WL95]. Ns-2 stellt für die Definition einer Szenariobeschreibung Klassen zur Erzeugung von Objekten für Knoten, Verbindungen und deren Eigenschaften bereit. Instanzen davon beschreiben dann ein Szenario. Eine Simulation in ns-2 wird dann durch Interpretation eines Szenarioskripts zur Laufzeit erreicht. Änderungen des Verhaltens der Simulationsumgebung zur Laufzeit werden durch ein Ereignissystem ermöglicht, das basierend auf Zeitstempeln Parameter und Komponenten eines Szenarios variiern kann.

Das an der University of Utah entwickelte System Emulab [WLS+02] verwendet zur Definition von Szenarien eine Teilmenge der ns-Syntax, die um spezifische Befehle erweitert wurde. Eine Szenariobeschreibung ist also analog zu ns-2 ein oTcl-Skript. Ein Szenario wird allerdings nicht erst zur Laufzeit interpretiert, sondern vorab durch einen Parser transformiert und die Komponenten und Eigenschaften einer Szenariobeschreibung zunächst in einer Datenbank abgelegt. Ein Szenario beschreibt dabei Ressourcenanforderungen, die zur Ausführung auf physisch vorhandene Ressourcen abgebildet werden. Ein Ereignissystem unterstützt die Änderung von Komponenten und deren Eigenschaften zur Laufzeit. Die Änderungen können durch Definition von Ereignissen vorab in einer Szenariobeschreibung bestimmt werden, oder spontan durch einen Benutzer zur Laufzeit abgesetzt werden.

Neben der Verwendung von Programmiersprachen bzw. Skriptsprachen können Szenarien auch in einem strukturierten Textformat, z.B. einem XML-basierten Format, beschrieben werden.

Für das NET-Projekt der Universität Stuttgart wird in [HLR02] eine Szenariobeschreibungssprache für die Durchführung von Emulationen vorgestellt, die auf XML (Extensible Markup Language) basiert. Die möglichen Sprachelemente sind in einer DTD (Document Type Definition) beschrieben. Zur Beschreibung eines Szenarios werden Sprachelemente zur Definition von Knoten, Verbindungen und deren Eigenschaften bereitgestellt. Um dynamische Änderungen einer Topologie im Verlauf einer Emulation vornehmen zu können, ist es möglich, die Verbindungseigenschaften einer Beschreibung zu einem bestimmten Zeitpunkt oder anhand von Sequenznummern von Paketen zu verändern. Durch verschiedene statistische Modelle (statistische Verteilungsfunktionen, Markov-Modelle) können die Parameter dabei flexi-

bel angepasst werden. Zur Spezifikation von Topologien wird eine graphische Oberfläche, der Szenario-Editor, bereitgestellt. Eine Szenariobeschreibung wird zur Durchführung dann einem Werkzeug zur Steuerung des Emulationsablaufs als Eingabedatei übergeben.

In [CEV03a] und [CEV03b] wird eine Szenariobeschreibungssprache vorgestellt, die ebenfalls auf XML basiert. Ziel der Entwicklung war es, eine werkzeugunabhängige Beschreibungssprache zu entwickeln, die nicht nur im Zusammenspiel mit einem bestimmten Simulations- oder Emulationswerkzeug eingesetzt werden kann. Dabei werden neben der Portabilität eine einfache Erweiterbarkeit und Möglichkeiten zur automatischen Umsetzung in werkzeugspezifische Beschreibungsformate als Entwicklungsziel genannt. Die Konstrukte zur Beschreibung eines Szenarios sind durch XML Schemas bestimmt. Als Basiskomponenten sind eine „Netzwerkbeschreibung“, eine „Verkehrsbeschreibung“ und eine Komponente zur Beschreibung von Kommandos für Simulations- bzw. Emulationswerkzeuge festgelegt. Die Beschreibung ist dabei werkzeugunabhängig. Daher muss zur Durchführung eines Experiments eine Szenariobeschreibung zunächst transformiert werden, so dass sie von einem Simulations- bzw. Emulationswerkzeug verwendet werden kann. Eine automatische Transformation erfolgt durch Regeln, die mittels XSLT (Extensible Stylesheet Language Transformation) implementiert werden. Es existieren zum Beispiel XSLT-Regelsätze, die eine Umsetzung der XML-Beschreibung in ein ns-2-kompatibles Format ermöglichen.

# Kapitel 4

# Experimente

In diesem Kapitel werden zunächst die Inhalte und Komponenten einer Szenariobeschreibung allgemein dargestellt. Die im Emulab-System verwendete Beschreibungssprache zur Definition von Szenarien wird dann anschließend näher betrachtet.

Emulab unterstützt bisher keine parametrisierten Experimentläufe, sondern nur die Durchführung einzelner Experimente. Für eine Erweiterung muss daher zunächst ein Verständnis über die Durchführung von einzelnen Experimenten vorhanden sein, um das System anschließend erweitern zu können. Daher werden in diesem Kapitel ebenfalls der Lebenszyklus und die Durchführung eines Experiments in Emulab detailliert betrachtet und Implementierungsdetails gegeben.

## 4.1 Szenarien

Ziel von Experimenten ist die Evaluation bestimmter Sachverhalte. Dazu wird von einem Benutzer ein bestimmter Kontex für ein Experiment festgelegt. Dieser Kontext wird *Szenario* genannt.

Ein Szenario beinhaltet alle beteiligten Komponenten, deren Eigenschaften sowie dynamische Aspekte, die während der Durchführung eines Experiments von Bedeutung sind. Dazu gehören z.B. Knoten, Netzwerkverbindungen oder Parameter für eine zu evaluierende Software. Das daraus resultierende Artefakt ist die *Szenariobeschreibung*.

Für Experimente im Umfeld von Netzwerkprotokollen und verteilten Systemen können die Inhalte einer Szenariobeschreibung wie in Abb. 4.1 dargestellt strukturiert werden. Die SUT (Software under Test) ist der zu untersuchende Gegenstand eines Experiments. Handelt es sich um ein Programm, müssen zumindest dessen Name und die zum Aufruf nötigen Parameter bekannt sein. Weiterhin kann es nötig sein, der SUT weitere Daten bereitstellen zu müssen, um einen sinnvollen Einsatz zu ermöglichen. Die Daten können direkt als Teil einer Szenariobeschreibung oder als Referenz auf eine Datei übergeben werden.

Die Durchführung eines Experiments, bzw. der Aufruf einer SUT, wird in einer Ausführungsumgebung vorgenommen, welche die Rahmenbedingungen für eine Untersuchung bilden. Bei Untersuchungen von Netzwerkprotokollen und verteilten Systemen sind die an einer Kommunikation beteiligten Knoten und deren Verbindungen untereinander von besonderer Be-

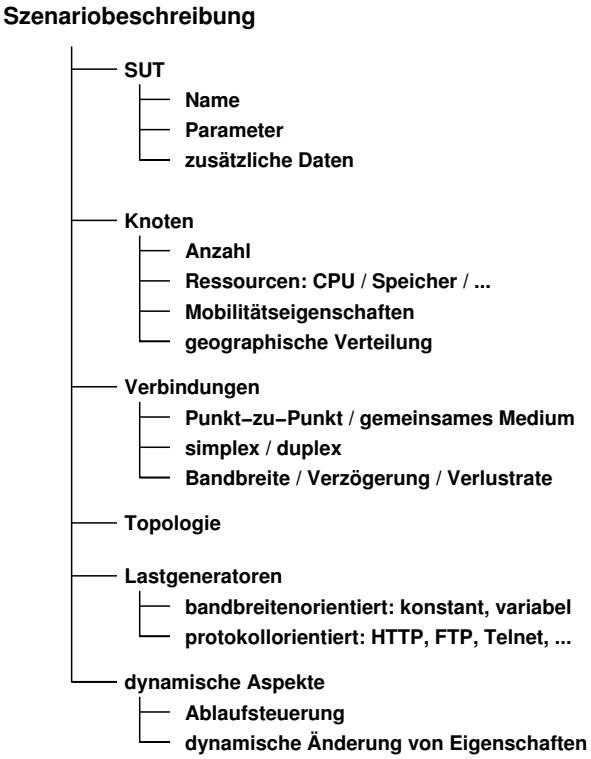



Abbildung 4.1: Inhalte einer Szenariobeschreibung

deutung.

Die Anzahl der an einem Experiment beteiligten Knoten wird meist durch die Topologie bestimmt. Neben der eigentlichen Anzahl, können Anforderungen an Knoten gestellt werden, die in einem Szenario durch eine Beschreibung ihrer Eigenschaften festgelegt wird. Dazu gehören z.B. Anforderungen an die Ausstattung der Knoten, wie verfügbare Rechenkapazität, Speichergrößen oder Bandbreiten von Netzwerkschnittstellen. Auch die geographische Verteilung kann durch entstehende Verzögerungen bei Übermittlung von Daten über mehrere Teilstrecken Einfluss auf Ergebnisse von Experimenten haben, so dass eine Angabe von Distanzen zwischen Knoten nötig sein kann.

Experimente in mobilen ad-hoc Netzen zeichnen sich zudem durch die Mobilität der beteiligten Kommunikationspartner aus. Für eine Durchführung solcher Experimente müssen in einer Szenariobeschreibung Mobilitätseigenschaften für Knoten definierbar sein. Eine solche Eigenschaft ist z.B. die Fortbewegungsgeschwindigkeit. Die Mobilität kann dann durch den Einsatz von Mobilitäts-Modellen umgesetzt werden, oder durch eine Festlegung von Wegpunkten, die Knoten während eines Experiments durchlaufen, nachgebildet werden. Eine Beschreibung der geographischen Verteilung wäre in diesem Zusammenhang dann einem Ausgangszustand gleichzusetzen, der die Standorte der Knoten vor der Ausführung eines Experiments festlegt.

Um eine Kommunikation zu ermöglich, müssen in einer Szenariobeschreibung Netzwerkverbindungen zwischen Knoten definiert werden. Als Typ der Verbindung kann meist zwischen Punkt-zu-Punkt Verbindungen oder Verbindungen mit gemeinsamem Medium, z.B. LANs, unterschieden werden, die über unidirektionale oder bidirektionale Kommunikationsfähigkeiten

verfügen können. Die Eigenschaften einer Verbindung haben meist einen großen Einfluss auf die Ergebnisse eines Experiments. Deshalb wird die Charakteristik bei Übertragungen durch eine Bestimmung der verfügbaren Bandbreite, Verzögerungen einzelner Pakete oder deren Verlustrate beeinflusst.

Die Topologie beschreibt die Architektur eines Netzwerks. Sie entsteht durch Zusammenschalten von Knoten über Verbindungen. Die Topologie wird daher meist implizit als Teil der Definition der beteiligten Komponenten (Knoten, Verbindungen) festgelegt.

Um realistische Rahmenbedingungen zu erhalten, werden, neben der eigentlichen Übermittlung von Daten durch die SUT, oft Hintergrundlasten benötigt. Die Hintergrundlast wird dabei durch Lastgeneratoren erzeugt. Die Generierung des zusätzlichen Netzwerkverkehrs kann aufgrund verschiedener Modelle erfolgen. Einfache Modelle sind bandbreitenorientiert und senden mit konstanter oder variabler Bitrate zwischen zwei oder mehreren Knoten. Eine weitere Möglichkeit besteht in der Generierung von Lasten durch Nachbilden von Kommunikationsmustern gängiger Protokolle, was damit einen protokollorientierten Ansatz darstellt.

Weiterhin sollten einem Experimentator Möglichkeiten zur Steuerung des Ablaufs und zur dynamischen Änderung der Eigenschaften der beteiligten Komponenten gegeben werden. So kann es während eines Experiments z.B. nötig sein, die Charakteristik einer Verbindung (z.B. die Verlustrate) zu ändern, um bei Funkübermittlung einen gestörten Kanal nachbilden zu können. Der Zeitpunkt einer Änderung kann dabei auf unterschiedliche Arten definiert werden. Zum Beispiel durch die Angabe eines Zeitpunkts relativ zum Startzeitpunkt des Experiments, oder durch Angabe von Sequenznummern von Paketen, ab deren Versand die Eigenschaften geändert werden.

## 4.2 Emulab Szenarien

Emulab stellt zur Durchführung von Experimenten eine Szenariobeschreibungssprache bereit, in der viele der im vorigen Abschnitt angesprochenen allgemeinen Inhalte umgesetzt werden können. Ein Szenario wird dazu in der Skriptsprache oTcl, einer objektorientierten Erweiterung von Tcl, beschrieben. Die Verwendung von oTcl für Szenariobeschreibungen ermöglicht es einem Experimentator, eigene Funktionen, Klassen und Methoden zu definieren.

Zur Definition der Komponenten eines Szenarios stellt Emulab eine Programmierschnittstelle bereit. Die verwendete Syntax ist kompatibel zu ns-2, allerdings werden nicht alle möglichen Kommandos von ns-2 übernommen, sondern nur eine Untermenge unterstützt. Die Szenariobeschreibungssprache ist aber um Emulab-spezifische Kommandos erweitert.

Im Folgenden wird zunächst die Beschreibungssprache von Emulab vorgestellt. Danach wird auf die Funktionsweise und Implementierung des Parsers eingegangen, der eine Szenariobeschreibung in ein intern von Emulab verwendetes Format umsetzt.

### Szenariobeschreibungen

Eine Szenariobeschreibung in Emulab enthält alle zur Durchführung eines Experiments benötigten Informationen. Dabei stellt eine Szenariobeschreibung eine virtuelle Repräsentation dar. Es werden also nicht physische vorhandene Ressourcen direkt referenziert, sondern Anforde-

rungen unabhängig von den physischen Ressourcen beschrieben. Die Aufgabe des Umsetzens auf physisch vorhandene Ressourcen bei der Ausführung wird in Abschnitt 4.3 betrachtet.

Emulab unterstützt verschiedene Ausführungsumgebungen für Experimente. Die Szenariobeschreibung eines Experiments ist davon nahezu unabhängig, wodurch es fast unverändert in mehreren Umgebungen durchgeführt werden kann.

Listing 4.1: oTcl-Skript des Internet Szenarios

```
1  set ns [new Simulator]          # Create the simulator
2  source tb_compat.tcl            # Add Emulab specific extensions
3
4  set bottleneck_bw       15Mbps
5  set client_bw           20Mbps
6
7  # add router
8  set leftRouter [$ns node]
9  set rightRouter [$ns node]
10
11 # create a bottleneck link
12 set bottleneck [$ns duplex-link $leftRouter $rightRouter $bottleneck_bw 0ms DropTail]
13
14 # add 3 client nodes to both sides and link to router
15 for {set i 0} {$i < 3} {incr i} {
16     set left$i [$ns node]
17     set leftLink$i [$ns duplex-link $leftRouter left$i $client_bw 0ms DropTail]
18
19     set right$i [$ns node]
20     set rightLink$i [$ns duplex-link $rightRouter right$i $client_bw 0ms DropTail]
21 }
22
23 $ns run          # "run" on Emulab
```

Eine Szenariobeschreibung wird in Emulab durch Anlegen und Verknüpfen von Objekten gebildet. Listing 4.1 zeigt das oTcl-Skript des in Abschnitt 2.1.3 vorgestellten Internet Szenarios. Das Internet Szenario verwendet eine „dumbbell“-Topologie und umfasst insgesamt 8 Knoten. Zwei Knoten dienen als Router. Zwischen ihnen besteht eine einzige Verbindung, die einen möglichen Flaschenhals bei der Kommunikation darstellt. An beiden Routern sind zur Überlastung des Flaschenhalses auf jeder Seite zudem jeweils drei weitere Knoten angebunden, die miteinander kommunizieren.

Anhand des Skripts in Listing 4.1 können die möglichen Inhalte einer Szenariobeschreibung und die Methode der Definition aufgezeigt werden. In einem Szenario muss zunächst ein Simulator-Objekt angelegt werden (Zeile 1). Um die Emulab-spezifischen Erweiterungen verfügbar zu machen, werden sie in einem Szenarioskript zunächst eingebunden (Zeile 2). Die wichtigsten Komponenten einer Szenariobeschreibung sind Knoten und deren Verbindungen untereinander. Neue Knoten werden über das Simulator-Objekt angelegt. In Zeile 8 wird ein Knoten angelegt, der im Internet Szenario als Router dient. Dazu wird ein neues Objekt durch Aufrufen der Methode „node“ des Simulator-Objekts angelegt und der entstandene Knoten da-

nach zusätzlich benannt. Der Name des Knotens wird bei der Durchführung des Experiments vom entsprechenden physischen Knoten übernommen.

Eine Verbindung zwischen Knoten wird, wie in Zeile 12 abgebildet, ebenfalls über das Simulator-Objekt angelegt. In diesem Fall wird die bidirektionale Punkt-zu-Punkt Verbindung zwischen den beiden Routern erstellt. Mit der Instanziierung wird dabei nicht nur der Typ der Verbindung bestimmt (duplex-link), sondern gleichzeitig auch die Eigenschaften der Verbindung. In diesem Fall wird eine Bandbreite von 15Mbps festgelegt, und Pakete werden nicht verzögert (0ms). Abgeschlossen wird eine Szenariobeschreibung durch Aufruf der „run“-Methode des Simulator-Objekts (Zeile 23). Dabei werden alle zu diesem Zeitpunkt erstellten Objekte vom Parser in einem XML-Format ausgegeben.

Das Internet Szenario schöpft allerdings nicht alle in Emulab verfügbaren Möglichkeiten aus. Für Knoten können z.B. weitere Eigenschaften bestimmt werden. Für einen Knoten kann festgelegt werden, welche Klasse von physischen Ressourcen zur Durchführung eines Experiments verwendet werden soll (tb-set-hardware). Die Angabe dieses Wertes beeinflusst daher die Wahl der Ausführungsumgebung, die physische Knoten („pc“), virtualisierte Knoten („pcvm“) oder WAN-Knoten („pcwa“) umfassen kann. Meist wird für alle Knoten eine Klasse angegeben. Eine Durchführung von Experimenten mit Knoten aus unterschiedlichen Klassen ist aber ebenfalls möglich.

Für Knoten besteht weiterhin die Möglichkeit, durch Angabe eines Festplattenabbildes das zu verwendende Betriebssystem festzulegen (tb-set-node-os). Eigene Programme können durch Angabe von Archiven oder Software-Paketen zur Laufzeit eines Experiments auf den Knoten installiert werden (tb-set-node-tarfiles, tb-set-node-rpms).

Die Eigenschaften von Netzwerkverbindungen werden bereits bei der Definition bestimmt, können aber auch nachträglich angepasst werden. Emulab unterstützt drei Eigenschaften, die zur Emulation eines Szenarios eingesetzt werden können: Bandbreite, Verzögerung und Verlustrate (tb-set-link-simplex-params). Die drei Eigenschaften können für beide Kommunikationsrichtungen unabhängig voneinander bestimmt werden.

Neben den beschriebenen Möglichkeiten für Knoten und Verbindungen werden von Emulab weitere Befehle bereitgestellt, z.B. Möglichkeiten zur Generierung von Hintergrundlast. Emulab unterstützt dazu verschiedene bandbreitenorientierte und protokollorientierte Modelle. Die Beschreibung erfolgt dabei durch so genannte „Agents“, die Quellen und Senken der Datenströme auf den Knoten definieren.

Zur Ausführung von Programmen auf Knoten können weitere Objekte angelegt werden. Dazu werden ein Programm und dessen Aufrufparameter spezifiziert und einem Knoten zugeordnet. Der Ausführungszeitpunkt hängt dann von der Art der Definition ab. Ein Programm kann z.B. direkt nach dem Start eines Experiments ausgeführt werden (tb-set-node-startcmd). Eine verzögerte Ausführung zu einem bestimmten Zeitpunkt ist durch ein Ereignissystem ebenfalls möglich.

Das Ereignissystem wird in Emulab zur dynamischen Steuerung eines Experiments verwendet. Kommandos und Änderungen der Eigenschaften eines Szenarios können damit zur Laufzeit stattfinden. Angegeben wird dazu ein Zeitpunkt relativ zum Beginn des Experiments sowie eine durchzuführende Aktion. Mögliche Aktionen sind z.B. das Ausführen von Programmen, die Änderung von Verbindungseigenschaften oder das Starten und Beenden von Lastgeneratoren. Dabei werden ebenfalls Aktionen auf Gruppen, z.B. einer Menge von Knoten, unterstützt.

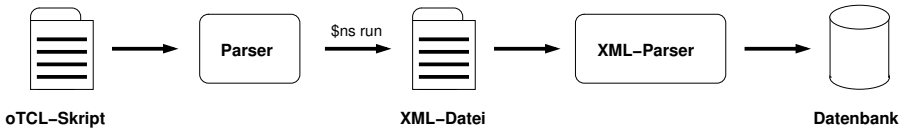



Abbildung 4.2: Transformationsprozess einer Szenariobeschreibung

Eine detaillierte Beschreibung aller Möglichkeiten und weitere Beispiele sind in der Dokumentation zu Emulab in [Gro05d], [Gro05c] und [Gro05b] verfügbar. Alle Emulab-spezifischen Erweiterungen werden in [Gro05e] beschrieben.

#### Transformationsprozess von Szenariobeschreibungen

Eine Szenariobeschreibung in Form eines oTcl-Skripts wird zur Durchführung eines Experiments nicht direkt verwendet, sondern beim Anlegen des Experiments zunächst in mehreren Stufen umgeformt. Der korrespondierende Prozess ist in Abb. 4.2 dargestellt. Ein oTcl-Skript wird zunächst durch einen oTcl-Parser (Parser) in eine XML-Darstellung transformiert, die seinerseits durch einen weiteren Parser (XML-Parser) in verschiedene Tabellen im Datenbank-Backend abgelegt wird.

Der oTcl-Parser von Emulab ist ebenfalls in oTcl implementiert. Die im vorigen Abschnitt angegebenen Möglichkeiten zur Beschreibung eines Experiments werden durch Bereitstellen von Klassen implementiert. Die Komponenten einer Szenariobeschreibung werden, wie beschrieben, durch Erzeugen von Objekten festgelegt und deren Verhalten und Eigenschaften durch Attribute implementiert. Ein Szenarioskript wird dann zur Transformation im oTcl-Parser interpretiert. Die Interpretation führt durch das Erstellen von Objekten zu einer internen Repräsentation im Parser. Die Definition eines Szenarioskripts wird durch eine „run“-Anweisung abgeschlossen. Alle zu diesem Zeitpunkt instanziierten Objekte und deren Attribute werden durch die Methode „run“ der Simulator-Klasse dann in einer XML-Darstellung ausgegeben.

Die XML-Datei wird in einem zweiten Schritt vom XML-Parser übernommen. Er liest die übergebene Datei ein und transformiert die Szenariobeschreibung dann, so dass sie in mehreren Tabellen abgelegt werden kann. Zur Durchführung von Experimenten werden dann von Emulab die Einträge in der Datenbank verwendet.

Das ursprüngliche oTcl-Skript wird allerdings ebenfalls in der Datenbank abgelegt, um Modifikationen von Experimenten für Benutzer einfach zu ermöglichen. Die zweistufige Umwandlung mit einer zwischenzeitlichen Darstellung in einer generischen Form in XML entkoppelt die verwendete Szenariobeschreibungssprache von der intern in Emulab verwendeten Darstellung eines Experiments. Andere Beschreibungssprachen könnten daher zu einem späteren Zeitpunkt ebenfalls unterstützt werden.

## 4.3 Emulab Experimente

Im Folgenden wird der Lebenszyklus eines Experiments und dessen Ablauf bei einer Durchführung näher betrachtet. Experimente werden in Emulab durch Kommandos eines Benutzers

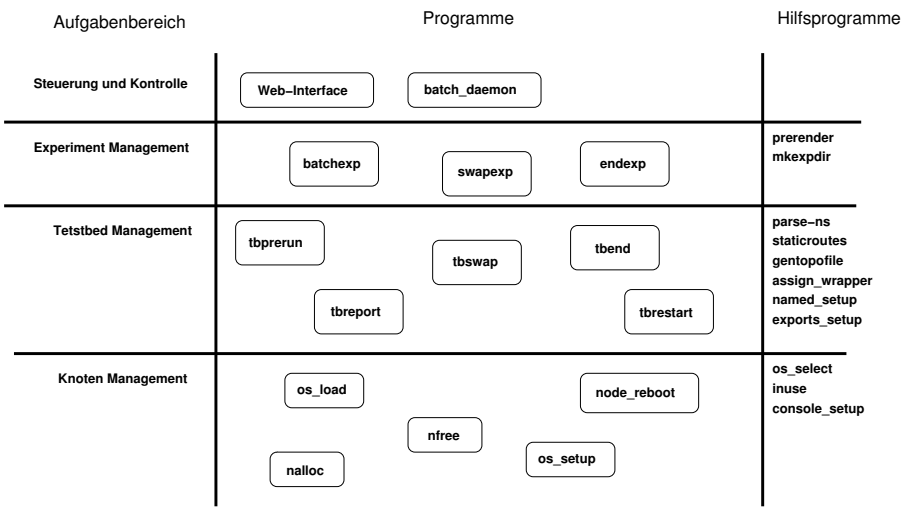



Abbildung 4.3: Architektur der Verwaltungsprogramme für Experimente (vereinfacht)

angelegt, durchgeführt oder entfernt. Die dazu benötigte Funktionalität ist in mehrere Programme aufgeteilt, die jeweils Teilaufgaben erfüllen. Aus dem Zusammenspiel der Programme lässt sich die in Abb. 4.3 dargestellte Architektur ableiten. Die beteiligten Programme können in vier Aufgabenbereiche eingeteilt werden. Alle Programme verwenden dabei Funktionalitäten der darunter liegenden Ebene. In der Darstellung sind nur die wichtigsten Programme enthalten. Zusätzlich werden aber ausgewählte Hilfsprogramme dargestellt, die innerhalb der jeweiligen Ebene verwendet werden.

Ein Benutzer steuert und kontrolliert Experimente durch Programme, die in der Darstellung in der obersten Ebene zu finden sind. Sie greifen über eine web-basierte Oberfläche auf das System zu, und setzen Kommandos ab, die zum Aufruf von Programmen der darunterliegenden Ebene führen. Für den Stapelbetrieb wird dazu eine Ablaufsteuerung (batch_daemon) realisiert. Der Stapelbetrieb kann daher als eine Art „virtueller Benutzer“ betrachtet werden, der Experimente automatisiert durchführt. Die Experimente im Stapelbetrieb werden im Folgenden auch als Batch-Experimente bezeichnet.

Die zweite Ebene (Experiment Management) beinhaltet Programme, die bei der Verwaltung von Experimenten verwendet werden. Dazu werden Programme zum Anlegen (batchexp), Aktivieren und Deaktivieren (swapexp) und Entfernen von Experimenten (endexp) bereitgestellt.

Die Programme verwenden Funktionalitäten der dritten Ebene, der „Testbed-Management“-Ebene. Die Programme dieser Ebene, dienen zur Verwaltung des Systems und Bereitstellung einer Ausführungsumgebung bei der Durchführung von Experimenten.

Auf der untersten Ebene sind Programme zu finden, die zur Verwaltung der physischen Knoten benötigt werden, z.B. für Reservierungen von Knoten (nalloc, nfree) oder das Aufspielen von Festplattenabbildern (os_load).

Für jedes Experiment wird in Emulab dessen Zustand gespeichert. Er gibt den aktuellen Status eines Experiments an. Die Zustandsübergänge werden durch eine Zustandsmaschine realisiert. Die Zustände, die ein Experiment einnehmen kann, sind, in leicht vereinfachter Form,

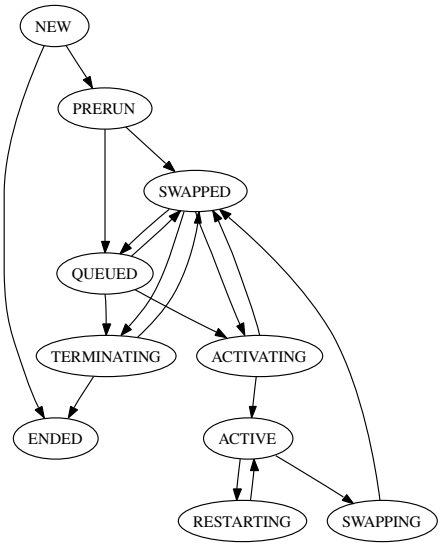



Abbildung 4.4: Zustandsgraph für Experimente (vereinfacht)

in Abb. 4.4 dargestellt. [1] Der Zustand eines Experiments wird von den Programmen der „Experiment Management“-Ebene verwaltet und geändert.

### Anlegen, Entfernen und Modifizieren

Ausgangspunkt für ein Experiment ist seine Szenariobeschreibung. Wie im vorigen Abschnitt geschildert, wird ein Szenario in Emulab durch ein oTcl-Skript beschrieben. Über eine webbasierte Oberfläche wird die Beschreibung dem System übergeben und das Experiment einem Projekt zugeordnet. Das Experiment befindet sich dann zunächst im Zustand NEW. Beim Anlegen (batchexp, tbprerun) werden dann verschiedene Aufgaben durchgeführt, während deren das Experiment im Zustand PRERUN ist. In dieser Phase findet z.B. das Parsen der Szenariobeschreibung statt, eine Visualisierung der Topologie des Experiments wird erstellt, statische Routen berechnet und Verzeichnisse, z.B. zur Speicherung von Log-Dateien, werden angelegt. Danach wird das Experiment in den Zustand SWAPPED gesetzt und kann durchgeführt werden.

Setzt ein Benutzer ein Kommando zum Entfernen eines Experiments ab, wird das Experiment in den Zustand TERMINATING gesetzt und nach Beendigung der Aktion in den Zustand ENDED, bevor es endgültig aus dem System gelöscht wird (endexp, tbend).

Modifikationen von Experimenten (swapexp), z.B. das Ändern der Topologie, sind im Zustand SWAPPED möglich oder während ein Experiment aktiv ist. Je nach Ausgangszustand werden dabei weitere Zustände durchlaufen, wie z.B. MODIFY_PARSE, MODIFY_RESWAP oder MODIFY_PRERUN.

[1] Zustände, die ein Experiment während der Modifikation durchläuft, sind aus Übersichtsgründen nicht enthalten.

### Betriebsarten und Ausführungszeitpunkte

Einem Benutzer stehen zwei Betriebsarten zur Auswahl. Im interaktiven Betrieb können Experimente durchgeführt werden, die ein direkte Interaktion des Benutzers voraussetzen. Mittels eines einfachen Stapelbetriebs können Experimente, die keine Interaktion der Benutzers während eines Experiments erfordern, ablaufen. Die Betriebsart wird bereits beim Anlegen eines Experiments bestimmt.

Ein interaktives Experiment wird vom Benutzer direkt gesteuert. Dazu gehört auch der Start- und Endzeitpunkt. Eine Anforderung zum Start wird dabei direkt vorgenommen und das System versucht unmittelbar die in der Szenariobeschreibung geforderten Ressourcen zu belegen. Sind nicht genügend Ressourcen verfügbar und der Versuch schlägt fehl, muss er zu einem späteren Zeitpunkt manuell wiederholt werden. Das Ende des Experiments wird ebenfalls vom Benutzer bestimmt und durch ein Kommando über die Web-Oberfläche gesteuert.

Der Startzeitpunkt von Batch-Experimenten wird dagegen durch die Ablaufsteuerung vom System bestimmt (batch_daemon). Ein Experiment wird zur Durchführung vom Benutzer lediglich in eine Warteschlange eingeordnet. Dazu wird es in den Zustand QUEUED überführt. Der weitere Ablauf wird dann von der Ablaufsteuerung übernommen. Emulab verfügt über keine Möglichkeit Prioritäten für Experimente festzulegen, so dass aus allen Experimenten im Zustand QUEUED alle 15 Sekunden in einem Round-Robin-Verfahren ein Experiment ausgewählt wird, für das dann ein Aktivierungsversuch unternommen wird. Sind für das Experiment nicht genügend Ressourcen vorhanden, so dass die Aktivierung fehlschlägt, wird der Versuch für dieses Experiment nach 15 Minuten wiederholt.

Mehrere Batch-Experimente können gleichzeitig aktiv sein. Um eine Verteilung der Ressourcenbelastung zu erreichen und damit auch den interaktiven Betrieb zu ermöglichen, wird die Anzahl der gleichzeitig aktiven Batch-Experimente jedoch begrenzt. Die Beschränkung wird mit Hilfe des Projektnamens eines Experiments und des Benutzernamens erreicht. Experimente eines Benutzers können nur gleichzeitig aktiv sein, wenn sie in verschiedenen Projekten angelegt wurden. Mehrere Experimente eines Projekts können gleichzeitig durchgeführt werden, wenn sie von verschiedenen Benutzern angelegt wurden. In jedem Projekt ist also zu einem Zeitpunkt maximal ein Experiment eines Benutzers aktiv.

Die Durchführung eines Batch-Experiments findet automatisch und ohne Interaktion des Benutzers statt. In der Szenariobeschreibung eines Experiments muss dazu für *jeden* Knoten ein Startkommando festgelegt werden, das nach der Aktivierung ausgeführt wird.

Ist ein Experiment erfolgreich aktiviert, prüft die Ablaufsteuerung periodisch alle 15 Sekunden, ob die Startkommandos auf den Knoten beendet wurden. Dazu wird der Rückgabewert der Startkommandos von den Knoten an den Steuerungsrechner übermittelt und in der Datenbank abgelegt. Ein Batch-Experiment ist dann beendet, wenn alle Knoten fehlerfrei konfiguriert werden konnten und die Werte der Startkommandos an das System übermittelt haben. In diesem Fall wird das Experiment dann von der Ablaufsteuerung deaktiviert.

Eine Mischform zwischen interaktivem Betrieb und Stapelbetrieb stellen Batch-Experimente dar, deren Ausführung nicht automatisiert ist. Ein solches Experiment wird als Batch-Experiment angelegt, es werden jedoch keine Startkommandos definiert. Da ein Benutzer über die Aktivierung eines Experiments benachrichtigt wird, stellt dies eine zeitversetzte Aktivierung dar. Damit können z.B. Experimente mit hohen Ressourcenanforderungen durchgeführt


Abbildung 4.5: Phasen eines Experiments

werden, deren Aktivierung ansonsten zu vielen manuellen Fehlversuchen führen würde.

Zusätzlich können beim Anlegen eines Experiments zwei weitere Attribute angegeben werden, die ebenfalls die Dauer eines Experiments beeinflussen [Gro05g]. Die *idle-time* bestimmt, wann ein Experiment als nicht mehr aktiv betrachtet wird. Zur Berechnung der aktuellen idle-time wird auf jedem Knoten ein Überwachungsprogramm gestartet, das Netzwerkschnittstellen und Terminals auf Aktivität prüft. Ist auf allen Knoten für einen längeren Zeitraum keine Aktivität zu erkennen, wird das Experiment beendet. Die *max-duration* bestimmt zusätzlich eine absolute, maximale Dauer eines Experiments. Bei Überschreiten dieser Zeit wird ein Experiment ebenfalls vom System beendet. Die Angabe der idle-time und max-duration ist allerdings nicht Pflicht, sondern kann von einem Benutzer optional angegeben werden. Gibt er diese Werte nicht an, muss er seine Entscheidung jedoch begründen. Analysen haben gezeigt, dass diese Form der kooperativen Zusammenarbeit durch freiwillige Beschränkung der Durchführungsdauer von Experimenten auch bei einer größeren Anzahl von Benutzern zu funktionieren scheint [ROLV06].

### Ablauf eines Experiments

In Abb. 4.5 ist der allgemeine Ablauf bei der Durchführung eines Experiments dargestellt. Er gliedert sich in drei Phasen. Die erste Phase dient dazu, ein Szenario in eine physische Repräsentation umzusetzen und dem Benutzer eine Ausführungsumgebung bereitzustellen. Dieser Vorgang wird vom System automatisch durchgeführt und in der Emulab-Terminologie als *Swapin* bezeichnet.

In der zweiten Phase, der Durchführungsphase, findet die eigentliche Ausführung eines Experiments statt. Bei Batch-Experimenten werden dabei die in der Szenariobeschreibung angegebenen Startkommandos ausgeführt, bei interaktiven Experimenten werden Aktionen vom Benutzer direkt vorgenommen.

Der Vorgang des *Swapout* bezeichnet die Endphase eines Experiments. Während dieser Phase werden alle belegten Ressourcen freigegeben und deren Konfiguration rückgängig gemacht, so dass sie als freie Ressourcen für andere Experimente wieder zur Verfügung stehen.

Ein Experiment durchläuft dabei mehrere Zustände. Ausgehend vom Zustand SWAPPED (bzw. QUEUED bei Batch-Experimenten) geht ein Experiment in den Zustand ACTIVATING über. Sind bei der Ressourcenbelegung und Konfiguration keine Fehler aufgetreten, wechselt das Experiment in den Zustand ACTIVE. Damit beginnt die Durchführungsphase. Für einen Swapout wird das Experiment in den Zustand SWAPPING überführt und beendet. Nach Durchführung ist das Experiment wieder im Zustand SWAPPED.

In der Swapin-Phase wird dem Benutzer, ausgehend von der Szenariobeschreibung, eine Ausführungsumgebung bereitgestellt (swapexp, tbswap). Dazu werden drei Aufgaben durch-

geführt:

- Umsetzen des virtuellen Szenarios auf physische Ressourcen,
- Belegung der Ressourcen und
- Konfiguration der Ausführungsumgebung.

Für die in der Datenbank gespeicherte Szenariobeschreibung muss zunächst eine Abbildung auf die physische Umgebung erfolgen. Ein komplexer Aspekt stellt dabei das Finden eines optimalen Reservierungschemas mit den zum Zeitpunkt des Swapins zur Verfügung stehenden Ressourcen dar. Dieses Problem ist in Abhängigkeit von den physischen Ressourcen, z.B. der Struktur des Verbindungsnetzwerks, zu lösen. Da es NP-hart ist, wird aus Effizienzgründen nur eine näherungsweise optimale Lösung berechnet. Das Programm „assign“ [RAL03] übernimmt diese Aufgabe mit Hilfe von „simulated annealing“. Ist eine mögliche Belegung gefunden, die alle Anforderungen des Experiments erfüllt, werden die benötigten Ressourcen reserviert (nalloc), anderenfalls wird der Swapin abgebrochen.

Nach der Reservierung der Ressourcen wird die Ausführungsumgebung konfiguriert. Dabei werden Konfigurationsänderungen zentral auf dem Steuerungsrechner und dezentral auf den beteiligten Knoten durchgeführt. Zentral werden DNS-Einträge zur Namensauflösung der in der Szenariobeschreibung angelegten Knoten erstellt (named_setup). Das Namensschema ist hierarchisch für alle Experimente eindeutig festgelegt. Weiterhin werden NFS-Freigaben konfiguriert, die das Projektverzeichnis und die Home-Verzeichnisse aller Projektmitglieder an die beteiligten Knoten exportiert (exports_setup). Die Bereitstellung der Topologie wird im Verbindungsnetzwerk durch virtual LANs (VLANs) erreicht. Dazu werden die Koppelelemente des Verbindungsnetzwerks entsprechend konfiguriert.

Vor dem Ausführen eines Experiments wird von Emulab sichergestellt, dass alle reservierten Knoten einen definierten Ausgangszustand besitzen. Da Emulab verschiedene Betriebssysteme für die Durchführung von Experimenten unterstützt, wird zu diesem Zeitpunkt das in der Szenariobeschreibung angegebene Betriebssystem durch Aufspielen eines Festplattenabbilds auf den Knoten installiert (os_load). Ist in der Szenariobeschreibung kein Betriebssystem angegeben, entfällt diese Aufgabe, da auf die Knoten ebenfalls *nach* jedem Experiment ein Standardfestplattenabbild aufgespielt wird. Für das Aufspielen von Festplattenabbildern wird in Emulab Frisbee eingesetzt [HSL+03]. Das Übermitteln und Aufspielen eines Festplattenabbilds erfolgt mit Frisbee in einem sehr kurzen Zeitraum. Die reine Übertragungszeit beträgt bei einer Größe des Abbildes von ca. 500 MB weniger als 2 Minuten und ist nahezu unabhängig von der Anzahl der beteiligten Knoten.

Ist das Betriebssystem aufgespielt, muss auf allen Knoten deren Konfiguration entsprechend der Szenariobeschreibung angepasst werden. Die Konfiguration wird dabei dezentral von den Knoten selbst gesteuert. Dazu wird ein Protokoll verwendet, das die Übertragung der dazu nötigen Informationen zwischen dem Steuerungsrechner und den Knoten festgelegt [Gro05h]. Alle Knoten stellen Anfragen zur Konfiguration, die ein Dienst auf dem Steuerungsrechner anhand der Informationen in der Datenbank beantwortet. Die Konfiguration der Knoten umfasst das Anlegen von Benutzerzugängen für die am Experiment beteiligten Benutzer, das Anpassen des Rechnernamens, die Konfiguration der Netzwerkschnittstellen, die Installation von zusätzlicher

Software und die Konfiguration der Eigenschaften der Netzwerkverbindungen entsprechend den Vorgaben der Szenariobeschreibung. Nach erfolgreicher Konfiguration aller Knoten wird dann zentral auf dem Steuerungsrechner als letzte Aktion der Swapin-Phase das Ereignissystem gestartet. Alle in einer Szenariobeschreibung festgelegten Ereignisse werden dann während der Durchführung zu den festgelegten Zeitpunkten an die Knoten gesendet und dort entsprechende Aktionen vorgenommen (z.B. Aktivieren und Deaktivieren von Netzwerkschnittstellen).

Sind beim Swapin keine Fehler aufgetreten, wird das Experiment in den Zustand ACTIVE gesetzt. Damit beginnt die Durchführungsphase. Bei interaktiven Experimenten werden dabei keine weiteren Aktionen vom System vorgenommen, bei Batch-Experimenten werden alle Startkommandos ausgeführt.

Ist das Experiment beendet, müssen alle Veränderungen rückgängig gemacht werden (swapexp, tbswap), um die Knoten wieder als freie Ressource dem System zur Verfügung zu stellen. Die in der Swapout-Phase durchzuführenden Aktionen sind dabei invers zu den während des Swapins vorgenommenen. Dabei wird zentral das Ereignissystem für das Experiment angehalten, VLANs bereinigt und DNS-Einträge und NFS-Freigaben wieder aus den Konfigurationsdateien des Steuerungsrechners entfernt. Alle reservierten Knoten werden dann wieder freigegeben. Sie werden allerdings nicht direkt als freie Ressource dem System wieder zugeführt, sondern es wird zunächst das Standardfestplattenabbild neu aufgespielt. Hierzu werden alle Knoten von einem Dienst auf dem Steuerungsrechner übernommen. Erst nach dem Aufspielen des Standardfestplattenabbilds werden die Knoten dann freigegeben (nfree) und stehen für weitere Experimente zur Verfügung.

# Kapitel 5

# Parametrisierbare Experimentläufe

Neben einzelnen isolierten Experimenten stellen parametrisierte Experimentläufe zusätzliche Funktionalität dar, die einen Experimentator bei der Durchführung von Experimenten zu Mess- und Evaluationszwecken unterstützen können.

Durch Evaluation verschiedener Randbedingungen können mit der Unterstützung von parametrisierbaren Experimentläufen verschiedene Situationen in einer integrierten Weise effizient untersucht werden. So ist bei der Entwicklung eines neuen Netzwerkprotokolls oder einer verteilten Anwendung deren Verhalten oftmals nicht immer vorab abzuschätzen. Um die Auswirkungen verschiedener Netzumgebungen analysieren zu können, ist daher oft eine große Anzahl von Experimenten durchzuführen, um schon vor dem eigentlichen Einsatz eines neuen Netzwerkprotokolls oder einer verteilten Anwendung deren Funktionalität sicherzustellen. Ebenfalls kann eine Auswertung mehrerer Implementierungsalternativen zu einem frühen Zeitpunkt der Entwicklungsphase durch eine Durchführung von mehreren Experimenten mittels Parametrisierung effizient erfolgen.

Eine Parametrisierung beschreibt im Kontext von Experimenten die Definition von Parametern und deren Werten, die eine automatisierte Durchführung von Experimenten unter verschiedenen Randbedingungen ermöglicht. Eine Parametrisierung führt in der Folge zu einer Wertemenge über die iteriert werden muss. Als Ergebnis entsteht eine Menge von Experimenten, die durchzuführen sind. Einzelne Experimente dieser Menge werden im Folgenden als Teilexperimente bezeichnet. Die gesamte Menge der Experimente ergibt sich aus den einzelnen Teilexperimenten und wird im Folgenden als Experimentlauf bezeichnet.

## 5.1 Anforderungen

In diesem Abschnitt werden zunächst allgemeine Anforderungen diskutiert, die an eine Unterstützung von parametrisierbaren Experimentläufen gestellt werden können.

Dazu werden Anforderungen in den Bereichen Benutzerschnittstelle, Funktionalität für Experimentläufe, deren Steuerung und der Verwaltung von Daten für Experimentläufe erörtert. Die Betrachtung findet dabei unabhängig von der späteren Zielumgebung statt.

### Benutzerschnittstelle

Ziel der Unterstützung von parametrisierbaren Experimentläufen ist es, die Durchführung einer Folge von Experimenten unter verschiedenen Randbedingungen zu ermöglichen.

Dabei werden von einem Experimentator verschiedene Aufgaben ausgeführt. Dazu gehören Verwaltungsaufgaben, wie z.B. Anlegen, Modifizieren und Entfernen von Experimenten, oder die Steuerung eines Experimentlaufs. Um solche Aufgaben erledigen zu können, muss dem Experimentator eine Schnittstelle zum System zur Verfügung gestellt werden. Der Umgang sollte so gestaltet sein, dass eine effiziente und effektive Arbeit mit dem System gewährleistet wird.

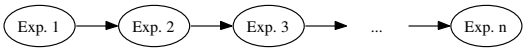
Experimente werden über eine Szenariobeschreibungssprache definiert und dem System übergeben. Die Definition einer Szenariobeschreibung sollte einfach verständlich, und die gewünschten Anforderungen an ein Szenario leicht umzusetzen sein. Dabei müssen einem Experimentator bei der Definition einer Szenariobeschreibung Freiheiten gegeben werden. Die Mächtigkeit und Allgemeinheit der Beschreibungssprache bestimmt also implizit die Menge der möglichen Szenarien. Für parametrisierte Experimentläufe müssen Möglichkeiten für die Definition von Parametern und deren Wertemenge bereitgestellt werden. Die Definition der Parameterwerte kann dabei intern, also innerhalb einer Szenariobeschreibung, erfolgen, oder durch Bereitstellen externer, unabhängig gespeicherter Daten. Eine leichte Änderbarkeit der Beschreibung ermöglicht eine Wiederholung unter geänderten Randbedingungen und unterstützt eine explorative Arbeitsweise bei vorab unvorhersehbaren Ergebnissen.

Ein Experimentlauf beinhaltet mehrere Teilexperimente. Neben dem Anlegen, Modifizieren und Entfernen von Experimenten, stellen Möglichkeiten zur Steuerung eine weitere Anforderung dar, die bei einzelnen Experimenten von geringerer Bedeutung sind. Insbesondere die Dauer eines kompletten Experimentlaufs ist gegenüber einzelnen Experimenten deutlich länger. Vor allem Möglichkeiten, einen Lauf zu unterbrechen und zu einem späteren Zeitpunkt fortzusetzen, müssen gegeben werden. Neben der Steuerung muss das System über Möglichkeiten verfügen, einem Experimentator Auskunft über den derzeitigen Fortschritt und Verlauf geben zu können. Dazu müssen Informationen über den Verlauf eines Experimentlaufs verfügbar sein.

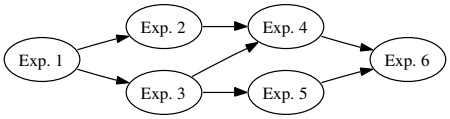
### Eigenschaften von Experimentläufen

Gegenüber einzelnen Experimenten weisen Experimentläufe zusätzliche Eigenschaften auf. So kann z.B. die Anzahl der Teilexperimente eines Laufs variieren. Je nachdem ob die Anzahl vorab festgelegt ist, kann zwischen zwei Ausprägungen unterschieden werden. Endliche Läufe werden durch eine endliche Wertemenge der Parameter begrenzt. Ein Experimentlauf besitzt demnach eine vorab bekannte, feste Anzahl von Teilexperimenten. Andererseits kann die Anzahl der Teilexperimente auch durch ein Abbruchkriterium oder eine maximal festgelegte Zahl begrenzt werden und die Anzahl daher vorab unbestimmt sein.

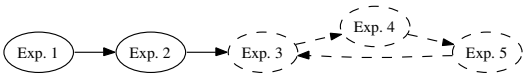
Ein Experimentlauf mit unbestimmter Anzahl der Teilexperimente ist z.B. bei der Suche eines Grenzwertes gegeben. So kann für eine verteilte Anwendung die Zahl der Kommunikationspartner deren Antwortzeiten beeinflussen. Soll die Antwortzeit einer Mindestanforderung genügen, wird diese als Grenzwert in einer Szenariobeschreibung für alle Teilexperimente fest-




(a) serielle Abhängigkeit




(b) komplexe Abhängigkeit




(c) zyklische Abhängigkeit

Abbildung 5.1: Abhängigkeitsgraphen für Experimentläufe

gelegt. Durch Verändern der Anzahl der Kommunikationspartner kann dann in einem Experimentlauf evaluiert werden, wann der Grenzwert mit steigender Zahl der Kommunikationspartner überschritten wird.

Die Parametrisierung stellt eine weitere Gestaltungsmöglichkeit einer Szenariobeschreibung dar. Daher ist zu entscheiden, welche Komponenten einer Parametrisierung unterzogen werden dürfen. Eine allgemeine Betrachtung ist aber kaum möglich. Der optimale Fall wäre es jedoch, alle Komponenten, die in Abschnitt 4.1 vorgestellt wurden, einer Parametrisierung unterziehen zu können. Für Emulab wird in Abschnitt 5.2 diskutiert, welche Komponenten eines Szenarios parametrisiert werden können.

Eine weitere Eigenschaft sind mögliche Abhängigkeiten zwischen Teilexperimenten eines Experimentlaufs. Derartige Abhängigkeiten entstehen genau dann, wenn ein Teilexperiment Ergebnisse eines anderen Teilexperiments benötigt, um durchgeführt werden zu können. Experimente sind also voneinander abhängig, wenn die erfolgreiche Durchführung eines oder mehrerer Experimente vorausgesetzt wird, um deren Ergebnisse zur Definition und Durchführung eines späteren verwenden zu können.

Ein Beispiel für diese Ausprägung ist ein Experimentlauf mit binärer Suche eines Optimums, wie die Suche nach einer optimalen Wartezeit in einem neuen Netzwerkprotokoll. Kann z.B. der Zustand eines Empfängers nicht vom Sender erkannt werden, wird nach einer fehlerhaften Übermittlung von Daten vor einem erneuten Senden zunächst gewartet. Die Wartezeit beeinflusst dabei allerdings meist auch die Effizienz der Kommunikation. Soll die Effizienz maximiert werden, und wird eine Verteilung der Messergebnisse in der Form erwartet, dass sich für die Wartezeit ein einziges Optimum ergibt, kann das Optimum durch binäre Suche mit Durchführung nur weniger Teilexperimente gefunden werden. Dabei kann ein folgendes Experiment allerdings erst durchgeführt werden, wenn das Ergebnis des vorigen Experiments bekannt ist.

Zu beachten ist, dass Abhängigkeiten zwischen Experimenten auch die Reihenfolge der Ausführung beeinflussen. In Abb. 5.1 sind verschiedene Abhängigkeitsgraphen dargestellt. Die Abhängigkeiten geben die Reihenfolge und den Datenfluss bei Durchführung eines Ex-

perimentlaufs wieder. In Abb. 5.1(a) besteht ein sequentieller Zusammenhang zwischen den Experimenten, und es müssen daher alle Experimente nacheinander durchgeführt werden. In Abb. 5.1(b) werden Ergebnisse aus den Experimenten 2 und 3 benötig, bevor das Experiment 4 durchgeführt werden kann. Es besteht allerdings auch die Gefahr, dass zyklische Abhängigkeiten entstehen, wie sie z.B. in Abb. 5.1(c) dargestellt sind. Experimentläufe mit zyklischen Abhängigkeiten können nur teilweise durchgeführt werden. Im Beispiel in Abb. 5.1(c) können nur die Experimente 1 und 2 durchgeführt werden. Eine Durchführung der Experimente 3–5 ist durch deren zyklische Abhängigkeit voneinander nicht möglich.

Ein Abhängigkeitsgraph ist durch die Definition eines parametrisierten Experimentlaufs zwar vorab vollständig festgelegt, die Möglichkeit zur Bestimmung des vollständigen Ablaufgraphen ist allerdings vorab nicht sichergestellt. So werden spätere Experimente eines Laufs im Allgemeinen durch die Ergebnisse eines vorigen beeinflusst. In Abb. 5.1(b) kann die Bestimmung des Ablaufgraphen z.B. erst möglich sein, nachdem Experiment 2 erfolgreich beendet wurde. Betrachtet man unendliche Läufe, kann der Ablaufgraph ebenfalls nicht vollständig vorab bestimmt werden, da die Definition eines Experiments unter Umständen erst nach Beenden des vorigen erfolgen kann. Nicht alle Teilexperimente eines Laufs werden dann vorab definiert, sondern durch die Ergebnisse eines vorigen festgelegt. Ein Ablaufgraph kann daher nur inkrementell bestimmt werden.

### Ablaufsteuerung von Experimentläufen

Ein Abhängigkeitsgraph hat auch Einfluss auf die Möglichkeit, mehrere Experimente eines Laufs parallel durchführen zu lassen. Dazu muss für die zeitgleiche Durchführung folgende Voraussetzung erfüllt sein: Zwei Experimente können parallel zueinander durchgeführt werden, wenn *keine* Abhängigkeit zwischen ihnen besteht. Da Experimente Ressourcen (z.B. Knoten) belegen, muss die Anzahl gleichzeitig laufender Teilexperimente durch Vorgaben des Systems beschränkt werden, um ein Zusammenspiel mehrerer Experimentläufe und interaktiver Experimente zu gewährleisten. Eine Ablaufsteuerung muss daher in der Lage sein, für parametrisierte Experimentläufe ein geeignetes Ablaufschema bereitzustellen.

Eine faire Verteilung verfügbarer Ressourcen und deren effiziente Auslastung sind ebenfalls Aufgaben der Ablaufsteuerung. Eine asynchrone Durchführung aufgrund der meist langen Laufzeit muss ebenfalls möglich sein.

Eine Ablaufsteuerung für Experimentläufe mit Beschränkung der Ressourcenauslastung ist allerdings eng mit der Forderung nach effizienter Unterstützung von Experimentläufen verbunden, weshalb sie in Kapitel 6 eingehender erörtert wird.

### Verwaltung von Experimentläufen

Zur Durchführung von Experimentläufen werden zusätzliche Informationen benötigt, die über die Informationen einzelner (Teil-)Experimente hinausgehen und für Experimentläufe verwaltet werden müssen.

Sollen Läufe unterbrochen und zu einem späteren Zeitpunkt fortgesetzt werden können, müssen Informationen über bereits durchgeführte Experimente gespeichert werden. Diese Aufgabe wird von einer Komponente zur Verwaltung des Experimenterfolgs übernommen. Ein

Experimenterfolg beschreibt dabei nicht den Erfolg im Sinne der Interpretation der Ergebnisse einer Messung, sondern lediglich, ob eine fehlerfreie Durchführung erfolgt ist.

Im Gegensatz dazu werden bei einer Ergebnisverwaltung (Mess-)Ergebnisse eines Experiments festgehalten. Für die Durchführung von Experimentläufen mit abhängigen Experimenten ist eine Speicherung der Ergebnisse Voraussetzung, um einen konkreten Ablauf festlegen zu können.

Eine Erweiterung ist die Bereitstellung von Ergebnissen bei mehreren Wiederholungen eines Experimentlaufs. Eine Speicherung der Historie von Ergebnissen bietet einem Experimentator weitere Unterstützung, z.B. für den Vergleich und die Validierung von Messergebnissen.

#### Fehlersemantik von Experimentläufen

Für die Durchführung von Experimentläufen ist eine Fehlersemantik zu definieren. Dabei können zwei Arten von Fehlern unterschieden werden. Fehler, die bei der Durchführung von Experimenten im System entstehen, und Fehler, die den Misserfolg eines Experiments beschreiben.

Fehler bei der Durchführung von Experimenten im System treten z.B. aufgrund mangelnder Ressourcen oder fehlerhafter Konfiguration bei Aktivierung eines Experiments auf. Es handelt sich also um Fehler, die bei der Verwaltung und Steuerung entstehen.

Fehler, die den Misserfolg eines Experiments beschreiben, müssen vom Experimentator an das System übermittelt werden. Er entscheidet aufgrund der Messergebnisse, ob ein Experiment fehlerhaft verlaufen ist.

Bei Durchführung von Experimentläufen müssen diese Fehler erkannt und nach Möglichkeit geeignete Maßnahmen für einen weiteren Ablauf getroffen werden. Mögliche Handlungsalternativen im Fehlerfall sind:

1. Den Experimentlauf unterbrechen, oder
2. bei unabhängigen Experimenten den Fehler ignorieren, oder
3. bei abhängigen Experimenten nur vom fehlgeschlagenen Experiment abhängige Experimente nicht ausführen.

In den ersten beiden Fällen sollten dem Experimentator Möglichkeiten zur Steuerung oder Wiederaufnahme des Experimentlaufs gegeben werden. Im dritten Fall, bei abhängigen Experimenten, ist keine Lösung möglich, und nur eine Teilmenge der Experimente kann durchgeführt werden.

## 5.2 Spezifikation

Aus den beschriebenen Anforderungen im vorigen Abschnitt wird im Zuge der Diplomarbeit eine prototypische Unterstützung für parametrisierbare Experimentläufe in Emulab entwickelt. Ziel ist es, die Durchführung parametrisierter Experimentläufe für Benutzer auf einfache Weise zu ermöglichen und in Emulab zu integrieren.

Die allgemeinen Anforderungen für parametrisierbare Experimentläufe, die im vorigen Abschnitt beschrieben wurden, werden aber zunächst nicht vollständig umgesetzt. Neben den in

Abschnitt 2.3 gegebenen Einschränkungen, die eine Durchführung von Experimenten in NET auf die Verwendung von physischen Knoten beschränkt, werden in der prototypischen Umsetzung zunächst keine abhängigen Experimente unterstützt. Daher ist auch eine Ergebnisverwaltung nicht zwingend erforderlich.

### 5.2.1 Stand in Emulab

Emulab implementiert bereits Funktionalitäten zur Verwaltung und Durchführung von Experimenten. Experimente werden jedoch lediglich als einzelne, isolierte Einheiten betrachtet. Zur Evaluation eines Sachverhaltes, z.B. einer Auswertung oder Analyse unter verschiedenen Randbedingungen, müssen daher mehrere Experimente manuell angelegt und durchgeführt werden. Eine Abbildung von mehreren Randbedingungen in einem einzigen Experiment im Sinne einer Parametrisierung ist nicht möglich. Weiterhin kann zwar die Durchführung einzelner Experimente durch einen Stapelbetrieb automatisiert werden, eine automatische Durchführung und Koordination mehrerer Experimente eines Laufs ist aber nicht vorgesehen. Eine gleichzeitige, mehrfache Durchführung eines Experiments ist ebenfalls nicht möglich.

### 5.2.2 Ansatz

Für eine Unterstützung von parametrisierbaren Experimentläufen in Emulab können verschiedene Ansätze gewählt werden, die sich durch die Integrationstiefe in das System unterscheiden. Diese können aus Programmen, die unabhängig von Emulab arbeiten, bis hin zu tief in das System eingreifende Änderungen bestehen.

Ein Ansatz ist z.B. das Bereitstellen von „Hilfsprogrammen“, die unabhängig von Emulab arbeiten. Aus einer parametrisierten Szenariobeschreibung wird dabei eine Menge einzelner Experimente erzeugt und der Ablauf über die Hilfsprogramme und die Kommandozeilen-Schnittstelle von Emulab gesteuert. Die Integrationstiefe ist dabei allerdings sehr gering und eine Steuerung über die web-basierte Oberfläche kaum möglich.

Eine weitere Möglichkeit parametrisierbare Experimentläufe zu unterstützen, ist es, eine Parametrisierung direkt vorzunehmen, indem die Parameterwerte einer Szenariobeschreibung für ein Experiment dynamisch vor der Durchführung geändert werden. Ein Experimentlauf wird dabei durch ein einziges, sich änderndes Experiment repräsentiert. Da ein Experiment in Emulab zu einem Zeitpunk nicht mehrfach aktiviert sein kann, ist aber eine Serialisierung bei der Durchführung des gesamten Experimentlaufs nicht zu verhindern. Eine derartige Umsetzung kann weiterhin nur durch tiefe Eingriffe in das System verwirklicht werden, und ist unter Umständen nur schwer zu handhaben.

Die Komplexität des gesamten Emulab-Systems und die Forderung nach einem in Emulab integrierten Umgang mit parametrisierten Experimentläufen legen es nahe, einen sinnvollen Mittelweg zwischen den beiden Extremen zu finden.

Ein Experimentlauf kann als eine Menge mehrerer, einzelner Experimente angesehen werden. Die Folge einer Parametrisierung ist bei dieser Betrachtungsweise die Menge aller Kombinationen, die sich aus den Werten aller Parameter ergibt. Variiert ein Szenario z.B. zwei Parameter, die jeweils drei Werte annehmen, kann der Experimentlauf durch insgesamt sechs

Teilexperimente beschrieben werden. Die Semantik, Experimente als „kleinste Arbeitseinheit“ zu betrachten, wird auf diese Weise für Benutzer beibehalten.

Dazu wird eine weitere Unterscheidung von Experimenten benötigt. Experimenten müssen weitere Attribute zugeordnet werden können, die zur Gliederung, Zusammenfassung und Verwaltung für parametrisierte Experimentläufe dienen können. Daher wird eine zusätzliche Semantik für Experimente definiert:

- Meta-Experimente stellen einen parametrisierten Experimentlauf dar, und
- Teilexperimente repräsentieren ein Experiment eines Experimentlaufs.

Ein *Meta-Experiment* (ME) ist die Abstraktion eines parametrisierten Experimentlaufs. Es beinhaltet alle Teilexperimente, die aufgrund einer Parametrisierung durchzuführen sind, und fasst sie in einer Art „Container“ zusammen. Ein *Teilexperiment* (TE) ist ein Experiment, das durch eine bestimmte Kombination der Werte aller Parameter gebildet wird.

Die Durchführung eines Experimentlaufs entspricht dann der Durchführung eines Meta-Experiments, durch das alle Teilexperimente durchgeführt werden. Das Meta-Experiment selbst wird dabei nicht als Experiment durchgeführt. Ein Benutzer kann somit über die Steuerung eines Meta-Experiments einen Experimentlauf durchführen. Das System übernimmt dabei die Koordination und Steuerung der Teilexperimente.

Da ein Experimentlauf bzw. Meta-Experiment automatisiert durchgeführt werden soll, ist dazu eine automatisierte Durchführung der Teilexperimente Voraussetzung. Emulab verfügt bereits über einen geeigneten Mechanismus, der Experimente in einem Stapelbetrieb automatisieren kann. Alle Teilexperimente stellen also eine weitere Form von Batch-Experimenten dar. Die Durchführung von Teilexperimenten kann daher wie bei „normalen“ Batch-Experimenten durch bereits bestehende Funktionalität von Emulab erfolgen. Teilexperimente unterscheiden sich von Batch-Experimenten durch eine erweiterte Ablaufsteuerung und zusätzlichen Eigenschaften.

### 5.2.3 Parametrisierung

Zur Unterstützung der Parametrisierung wird eine Möglichkeit benötig, in einer Szenariobeschreibung eines Meta-Experiments Parameter und deren Werte definieren zu können, um daraus nicht-parametrisierte Szenariobeschreibungen für alle Teilexperimente erstellen zu können.

#### Parametrisierte Szenariobeschreibungen

Das Verwenden der Skriptsprache oTcl mit ns-spezifischen Erweiterungen zur Definition einer Szenariobeschreibung bietet bereits Möglichkeiten, parametrisierte Experimentläufe zu unterstützen. Parameter eines Experimentlaufs können als Variablen angelegt werden. Durch das Verwenden von z.B. Schleifenkonstrukten können dann die möglichen Kombinationen der Parameterwerte abgeleitet werden.

Die Definition eines Experimentlaufs wird daher durch eine automatische Generierung von Szenariobeschreibungen für Teilexperimente ermöglicht, die eine bestimme Ausprägung der Parameterwerte darstellen. Dazu ist es nötig, aus einer parametrisierten Szenariobeschreibung

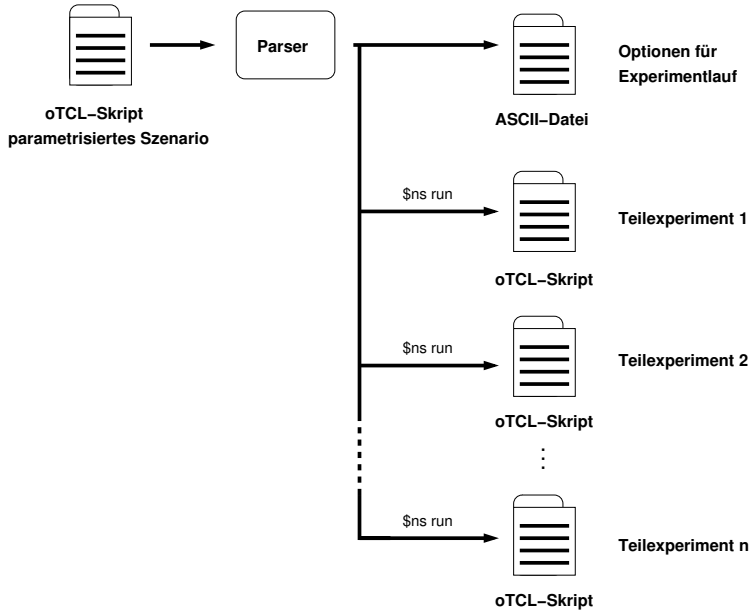



Abbildung 5.2: Transformationsprozess einer parametrisierten Szenariobeschreibung

alle Szenariobeschreibungen der Teilexperimente zu generieren. Der Prozess ist in Abb. 5.2 dargestellt. Der Parser von Emulab wird daher erweitert, so dass eine Transformation einer Szenariobeschreibung von oTcl nach oTcl möglich ist, die aus den Parametern und deren Werten nicht-parametrisierte Szenariobeschreibungen erzeugt.

Durch Definition von Objekten und deren Eigenschaften werden im Parser intern Objekte instanziiert und untereinander verknüpft. Durch den Aufruf der Methode „run“ des Simulator-Objekts wird eine Beschreibung abgeschlossen (siehe Abschnitt 4.2). Ein Experimentlauf wird durch ein parametrisiertes oTcl-Skript beschrieben, das durch mehrmaligen Aufruf der „run“-Methode zu mehreren Ausprägungen führt. Bei jedem „run“-Aufruf wird dabei aus der aktuellen internen Beschreibung ein neues oTcl-Skript erzeugt, das die Szenariobeschreibung für ein Teilexperiment darstellt. Zusätzlich werden Informationen, die den gesamten Experimentlauf beeinflussen, in einer weiteren Datei abgelegt.

Ein solcher Ansatz bietet viele Vorteile. Der Einsatz der Skriptsprache ermöglicht die Verwendung von Funktionen, Schleifen und bedingten Verzweigungen, mit deren Hilfe eine übersichtliche und flexible Definition erfolgen kann. Durch optionalen Einsatz von Bibliotheken mit allgemeinen Komponenten, können Teile einer Definition für mehrere Experimentläufe wiederverwendet werden. Weiterhin wird die Programmierschnittstelle nahezu unverändert übernommen, und eine Parametrisierung ist für Benutzer ohne intensiven Lernaufwand möglich. Bereits bestehende Szenariobeschreibungen können auch einfach für eine Parametrisierung erweitert werden.

Ein Nachteil des Ansatzes ergibt sich daraus, dass vom Parser nicht festgestellt werden kann, welche Komponenten eines Szenarios von der Parametrisierung betroffen sind. Parameter werden implizit durch Definition von Variablen erzeugt, dabei aber nicht gekennzeichnet. Damit ist in einer parametrisierten Szenariobeschreibung nicht ohne weiteres erkennbar, welche Komponenten im Verlauf des Parsens geändert werden. Als Folge muss die Definition eines

Szenarios für jedes Teilexperiment immer vollständig erfolgen.

Für eine übersichtliche Gestaltung der Skripte wird jedoch eine Ausnahme festgelegt: Das Simulator-Objekt wird nach jeder Erzeugung eines Teilexperiments automatisch neu angelegt, und muss daher nur einmal zu Beginn definiert werden.

### Parametrisierbare Komponenten

Eine Szenariobeschreibung umfasst, wie bereits in Abschnitt 4.2 dargelegt, verschiedene Komponenten. Durch die Umsetzung der Parametrisierung in mehrere Teilexperimente, können alle Komponenten einer Szenariobeschreibung parametrisiert, und nahezu alle Emulab-spezifischen Erweiterung verwendet werden.

Die umfasst grundlegende Komponenten, wie Menge und Eigenschaften von Knoten, die Topologie, oder Eigenschaften von Verbindungen wie Bandbreite oder Verzögerung. Ebenso können Parameter der Testsubjekte, Hintergrundlasten oder dynamische Aspekte variiert werden. Auch Eigenschaften, die Auswirkungen auf physische Ressourcen haben, können durch Parameter beeinflusst werden. Das beinhaltet z.B. Angaben über zu verwendende Betriebssysteme oder zu installierende Softwarepakete.

Zu Testzwecken kann die Abbildung von Knoten der Szenariobeschreibung auf bestimmte physische Knoten von einem Experimentator manuell vorgenommen werden (tb-set-fixed). Die Verwendung des Befehls ist für parametrisierte Experimentläufe nicht unterbunden, eine Aktivierung von Teilexperimenten mit festgelegter Auswahl der physischen Knoten wird aber fehlschlagen, wenn die gewählten physischen Knoten bereits von einem anderen (Teil-)Experiment reserviert wurden. In diesem Fall wird der Experimentlauf angehalten.

### Erweiterungen der Programmierschnittstelle

| Befehl | mögliche Werte | Standardwert |
|---|---|---|
| tb-exprun-expfailure | ignore \| fatal | ignore |
| tb-exprun-onreclaim | reconfigure \| reboot \| reload | reconfigure |
| tb-exprun-priority | low \| med \| high \| urgent | med |

Tabelle 5.1: Zusätzliche Befehle für Experimentläufe

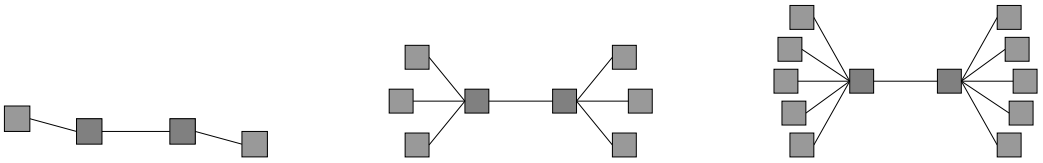
Die Befehle der bestehenden Programmierschnittstelle können zur Beschreibung von Teilexperimenten verwendet werden. Um das Verhalten eines gesamten Experimentlaufs beeinflussen zu können, werden jedoch Erweiterungen benötigt. Da sie zur Festlegung von Eigenschaften des gesamten Experimentlaufs dienen, werden sie in einer Szenariobeschreibung nur einmal (für das Meta-Experiment) definiert.

In Tab. 5.1 sind diese Erweiterungen aufgeführt. Die neuen Befehle dienen zur Definition einer Fehlersemantik (tb-exprun-expfailure), effizienter Ressourcennutzung (tb-exprun-onreclaim), und Koordination mehrerer Experimentläufe (tb-exprun-priority). Eine detaillierte Beschreibung der Befehle und deren Auswirkungen werden in den entsprechenden Abschnitten gegeben.

**Beispiel: Internet Szenario**

Als Beispiel wird das in Abschnitt 2.1.3 vorgestellte Internet Szenario parametrisiert. Das Szenario soll die Evaluation des Verhaltens einer verteilten Anwendung bei Überlast im Netz ermöglichen.

Um eine Evaluation unter verschiedenen Randbedingungen zu ermöglichen, wird das Szenario durch zwei Parameter variiert. Einerseits wird eine Änderung der Topologie vorgenommen, indem die Anzahl der an die beiden Router angebundenen Knoten verändert wird. Auf jeder Seite werden jeweils ein, drei oder fünf Knoten angebunden. Andererseits wird durch einen zweiten Parameter die Bandbreite des Flaschenhalses zwischen den Routern variiert. Dazu werden Werte von 10 Mbps, 15 Mbps und 20 Mbps verwendet. Über die möglichen Kombinationen der drei Werte jeden Parameters entstehen damit insgesamt neun Teilexperimente mit drei Topologien, die in Abb. 5.3 dargestellt sind.



(a) 4 Knoten (b) 8 Knoten (c) 12 Knoten

Abbildung 5.3: Topologien des parametrisierten Internet Szenarios

Zur Beschreibung der Teilexperimente wird das oTcl-Skript aus Abschnitt 4.2 erweitert. Das parametrisierte Szenarioskript ist in Listing 5.1 abgebildet. Im Skript wird zunächst wieder ein Simulator-Objekt angelegt und die Emulab-spezifischen Erweiterungen eingebunden (Zeile 1–2). Darauf folgen die Definition der Eigenschaften des Experimentlaufs (Zeile 4–5), und in den Zeilen 7–10 die Definition der Parameter und deren Wertemengen (min_nodes, max_nodes, bottleneck_bw).

Durch die beiden Schleifen (Zeilen 39–44) werden für ein Teilexperiment die Parameter variiert. Die Funktion „make-topo“ (Zeilen 16–36) erstellt aus den aktuellen Werten der Parameter jeweils eine vollständige Szenariobeschreibung. Die Erstellung der Teilexperimente wird durch mehrmaliges Aufrufen der Methode „run“ angestoßen (Zeile 42). Dazu wird vor jedem Aufruf das gesamte Szenario mit Hilfe der Funktion „make-topo“ angelegt.

Zu beachten ist auch, dass in der Beschreibung für jeden Knoten ein Startkommando definiert wird (tb-set-node-startcmd). Für eine automatische Durchführung der Teilexperimente ist das Voraussetzung. Durch die Art der Implementierung des Emulab-Parsers müssen außerdem alle Objekte in globalem Kontext angelegt werden. Deshalb wird zu Beginn von „make-topo“ ein „uplevel“-Kommando eingefügt.

### 5.2.4 Experimentläufe

Für einzelne Experimente sind bereits Funktionalitäten zur Verwaltung und Steuerung in Emulab vorhanden. Die vorhandene Funktionalität wird nach Möglichkeit für die Unterstützung

Listing 5.1: oTcl-Skript des parametrisierten Internet Szenarios

```
1  set ns [new Simulator]
2  source tb_compat.tcl
3
4  tb-exprun-expfailure     ignore
5  tb-exprun-onreclaim      reconfigure
6
7  set min_nodes            1
8  set max_nodes            5
9  set node_step            2
10 set bottleneck_bw        [ list 10Mbps 15Mbps 20Mbps]
11
12 set client_bw            20Mbps
13 set exp_duration         120
14
15 # - - - - - - - -
16 proc make-topo {} { uplevel #0 {
17   # add router
18   set leftRouter [$ns node]
19   set rightRouter [$ns node]
20   tb-set-node-startcmd leftRouter "~/scripts/iNET-SUT.sh $exp_duration"
21   tb-set-node-startcmd rightRouter "/bin/true"
22
23   # create bottleneck link
24   set bottleneck [$ns duplex-link $leftRouter $rightRouter $this_bw 0ms DropTail]
25
26   # add clients to both sides
27   for {set i 0} {$i < $this_nodes} {incr i} {
28       set tmpL [set left$i [$ns node]]
29       tb-set-node-startcmd $tmpL "/bin/true"
30       set tmpLL [set leftL$i [$ns duplex-link $leftRouter $tmpL $client_bw 0ms DropTail] ]
31
32       set tmpR [set right$i [$ns node]]
33       tb-set-node-startcmd $tmpR "/bin/true"
34       set tmpRL [set rightL$i [$ns duplex-link $rightRouter $tmpR $client_bw 0ms DropTail] ]
35   }
36 }}
37
38 # - - - - - - - - -
39 for {set this_nodes $min_nodes} {$this_nodes <= $max_nodes} {incr this_nodes $node_step} {
40   foreach this_bw $bottleneck_bw {
41       make-topo
42       $ns run
43   }
44 }
```

von parametrisierten Experimentläufen abgebildet und zusätzlich um geeignete Mechanismen erweitert.

### Funktionalität

Die Arbeit mit Emulab wird über eine web-basierte Oberfläche möglich, in der Kommandos an das System abgesetzt werden können, die zu Operationen auf Experimenten führen. Im Folgenden werden die für parametrisierte Experimentläufe verfügbaren Kommandos beschrieben.

Parametrisierte Experimentläufe werden analog zu einzelnen Experimenten durch Übergabe einer Szenariobeschreibung in Form eines oTcl-Skritps dem System übergeben und angelegt. Hiezu werden nach der Übergabe der Szenariobeschreibung ein Meta-Experiment angelegt und alle Teilexperimente generiert.

Alle weiteren Kommandos können dann danach klassifiziert werden, ob sie eine Operation auf dem Meta-Experiment oder einzelnen Teilexperimenten durchführen.

Die Steuerung eines Experimentlaufs erfolgt über das Meta-Experiment. Demzufolge sind Operationen wie Starten, Anhalten, und Entfernen jeweils nur für das Meta-Experiment zugelassen. Eine Durchführung der Operationen auf einzelnen Teilexperimenten ist nicht möglich.

Das Starten eines Experimentlaufs wird über ein Kommando auf das Meta-Experiment durchgeführt. Wird ein Meta-Experiment gestartet, führt das zur Durchführung aller Teilexperimente. Einzelne Teilexperimente können nicht unabhängig vom Meta-Experiment durchgeführt werden.

Da Experimentläufe durch eine Menge von Teilexperimenten gebildet werden, kann eine gegenüber einzelnen Experimenten längere Laufzeit vermutet werden. Das Anhalten und eine spätere Fortsetzung eines Experimentlaufs stellen daher eine wichtige Funktionalität dar. Das Anhalten wird durch zwei Kommandos ermöglicht. Ein Experimentlauf kann unterbrochen oder pausiert werden. Ein aktives Meta-Experiment wird unterbrochen, indem alle aktiven Teilexperimente abgebrochen werden. Aktive Teilexperimente werden dabei nicht zu Ende geführt, sondern umgehend beendet. Die zweite Möglichkeit einen Experimentlauf anzuhalten ist das Pausieren. Das Pausieren eines Meta-Experiments führt lediglich dazu, dass keine weiteren Teilexperimente neu gestartet werden. Bereits aktive Experimente werden jedoch nicht unterbrochen, sondern zu Ende geführt. Die Operation ist also erst nach der normalen Beendigung aller zum Zeitpunkt des Kommandos aktiven Experimente abgeschlossen. Ein Anhalten einzelner Teilexperimente unabhängig vom Meta-Experiment wird nicht unterstützt.

Ein Kommando zum Entfernen wird ebenfalls bereitgestellt und ist nur für Meta-Experimente erlaubt. Teilexperimente können nicht unabhängig vom Meta-Experiment entfernt werden.

Die Modifikation parametrisierter Experimentläufe wird derzeit nicht unterstützt. Da sich dabei die Szenariobeschreibung aller Teilexperimente ändert, ist ein Neuanlegen der Teilexperimente nötig. In vielen Fällen wird sich aber zusätzlich die Anzahl der Teilexperimente ändern. Für einen allgemeinen Ansatz ist es daher nötig, alle Teilexperimente vor der Modifikation des Meta-Experiments zu löschen und nach der Modifikation neu anzulegen, was daher zum gleichen Aufwand wie ein Neuanlegen des gesamten Experiments führt.

Weitere in Emulab bereits verfügbare Kommandos dienen meist zu Informationszwecken, wie z.B. die Ansicht von Log-Dateien, der Szenarioskripte oder der Visualisierung und detail-

lierten Darstellung von Experimenten. Die Kommandos können dementsprechend für Meta-Experimente und einzelne Teilexperimente durchgeführt werden.

### Ablaufsteuerung

Zur Unterstützung von parametrisierbaren Experimentläufen wird eine Ablaufsteuerung benötigt, die zwei Aufgaben erfüllen muss: Die Koordination von Meta-Experimenten und die Steuerung und Koordination von Teilexperimenten. Die Aufgaben werden von einer Experimentlauf-Steuerung übernommen.

Gesteuert werden Meta-Experimente vom Benutzer. Die Koordination, z.B. eine Ausführungsreihenfolge verschiedener Meta-Experimente, wird von der Experimentlauf-Steuerung übernommen. Die Koordination von Teilexperimenten, z.B. eine geeignete Auswahl und deren Reihenfolge, sowie die Steuerung der Teilexperimente, um deren automatische Durchführung zu ermöglichen, wird ebenfalls durch die Experimentlauf-Steuerung vorgenommen.

Teilexperimente benötigen zur Durchführung Ressourcen, die vom System ausgewählt und bereitgestellt werden müssen. Die Auswahl und Reihenfolge muss dabei durch ein geeignetes Schema unterstützt werden. Das Schema wird allerdings erst in Kapitel 6 im Zusammenhang mit einer effizienten Unterstützung von Experimentläufen erläutert und zunächst lediglich festgehalten, dass eine Experimentlauf-Steuerung diese Aufgaben übernimmt.

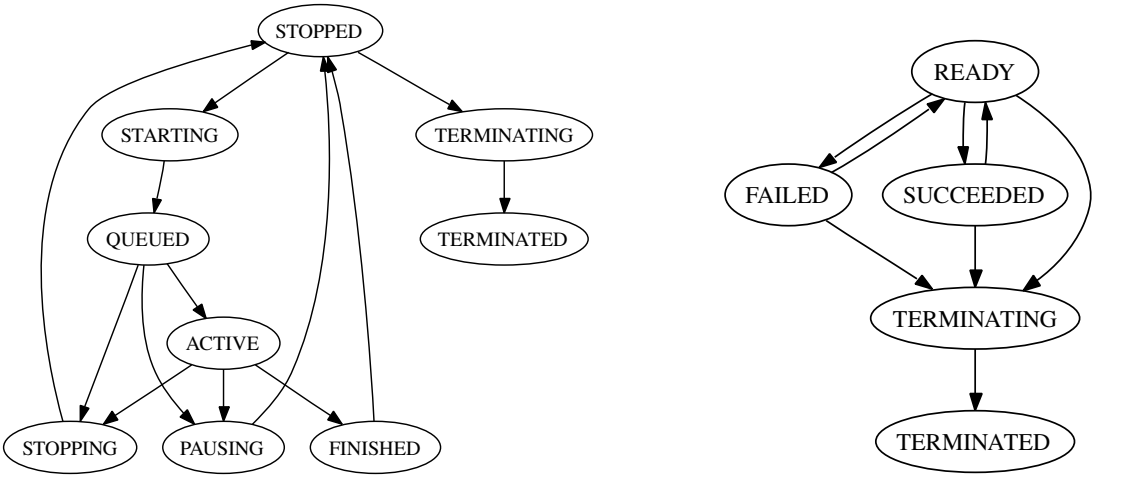
### Ablauf eines Meta-Experiments

Ein Meta-Experiment dient lediglich als Container für Teilexperimente und wird selbst nicht als Experiment ausgeführt. Daher kann der Zustand, wie ihn Emulab verwaltet (siehe Abschnitt 4.3), für eine Steuerung der Teilexperimente nicht verwendet werden. Eine weitere Zustandsmaschine wird benötigt. Sie umfasst die im Zustandsgraph in Abb. 5.4(a) dargestellten Zustände und Übergänge. Der Zustand wird zur besseren Unterscheidung im Folgenden als ME-Zustand bezeichnet, der in Emulab bereits existierende als Emulab-Zustand.

Für Teilexperimente wird zusätzlich ebenfalls ein weiterer Zustandsgraph definiert, der als TE-Zustand bezeichnet wird und für jedes einzelne Teilexperiment verwaltet wird (Abb. 5.4(b)). Da Teilexperimente annähernd Batch-Experimente darstellen, wird zu deren Durchführung weiterhin der Emulab-Zustand verwendet. Der TE-Zustand dient lediglich zu Verwaltungszwecken im Kontext des Meta-Experiments und wird daher nicht zur eigentlichen Durchführung benötigt.

Im Folgenden wird die Durchführung eines Meta-Experiments mit allen Teilexperimenten beschrieben und die neu eingeführten Zustandsmaschinen näher betrachtet. Nach dem Anlegen ist ein Meta-Experiment zunächst in Zustand STOPPED und alle Teilexperimente im Zustand READY. Ein Kommando zur Ausführung durch einen Benutzer führt zum Übergang des Meta-Experiments in den Zustand STARTING. Die Experimentlauf-Steuerung übernimmt dann den weiteren Ablauf und führt das Meta-Experiment dazu zunächst in den Zustand QUEUED über.

Um Teilexperimente durchführen zu können, müssen genügend Ressourcen verfügbar sein. Sind genügend Ressourcen vorhanden, wird das Meta-Experiment aktiviert (ACTIVE), und die Durchführung der Teilexperimente beginnt. Dabei wird für jedes Teilexperiment der in Abschnitt 4.3 beschriebene Ablauf durchgeführt.




(a) ME-Zustände für Meta-Experimente

(b) TE-Zustände für Teilexperimente

Abbildung 5.4: Zustandsgraphen für Meta- und Teilexperimente

Der TE-Zustand eines jeden Teilexperiments ist dabei vor der Durchführung eines Meta-Experiments zunächst READY. Nach Beendigung eines Teilexperiments wird er auf SUCCEEDED oder FAILED gesetzt. Damit ist ersichtlich, dass das Teilexperiment durchgeführt wurde.

Während der Durchführung kann ein Benutzer den Experimentlauf unterbrechen oder pausieren. Eine Wiederaufnahme zur Durchführung der restlichen Teilexperimente ist dann möglich, da aus dem Zustand der Teilexperimente (TE-Zustand) hervorgeht, welche bereits durchgeführt wurden.

Nach Beendigung eines Experimentlaufs befinden sich alle Teilexperimente im Zustand SUCCEEDED oder FAILED und das Meta-Experiment im Zustand FINISHED. Für eine erneute Durchführung des gesamten Experimentlaufs müssen Meta- und Teilexperimente daher zunächst zurückgesetzt werden. Dazu wird eine Operation bereitgestellt, die alle Teilexperimente von SUCCEDED oder FAILED in den Zustand READY überführt, und das Meta-Experiment von FINISHED auf STOPPED setzt.

Die noch nicht beschriebenen Zustände TERMINATING und TERMINATED beider Zustandmaschinen werden beim Entfernen durchlaufen. Wird ein Meta-Experiment entfernt, wird diese Operation ebenfalls auf alle Teilexperimente angewandt. Nach Absetzen des Kommandos zur Entfernung geht ein Meta-Experiment in den Zustand TERMINATING. Daraufhin werden alle Teilexperimente gelöscht. Sie nehmen dabei ebenfalls den Zustand TERMINATING ein, und nach erfolgreichem Entfernen den Zustand TERMINATED. Danach wird das Meta-Experiment entfernt und in den Zustand TERMINATED versetzt, bevor es endgültig aus dem System gelöscht wird.

### Erfolgsverwaltung

Ein Teilexperiment ist beendet, wenn alle physischen Knoten die in der Szenariobeschreibung für jeden Knoten definierten Startkommandos ausgeführt haben, und deren Rückgabewerte an das System gemeldet wurden. Als erfolgreich wird ein Teilexperiment dann angesehen, wenn

alle Startkommandos erfolgreich beendet wurden.

Da allerdings nach Ablauf eines Experiments die Rückgabewerte bei Freigabe der physischen Knoten wieder gelöscht werden, ist eine Speicherung über die Laufzeit eines Experiments hinaus zunächst nicht gegeben. Um für Teilexperimente eine Erfolgsverwaltung und -kontrolle zu ermöglichen, muss das Ergebnis daher an anderer Stelle gespeichert werden.

Da nur der Erfolg eines Teilexperiments von Interesse ist, werden nicht die Rückgabewerte der Startkommandos gespeichert, sondern der Erfolg des Teilexperiments. Dazu wird der TE-Zustand verwendet. Nach Beendigung eines Teilexperiments wird der Erfolg im TE-Zustand festgehalten. War das Experiment erfolgreich (alle Startkommandos lieferten 0) geht der Zustand in SUCCEEDED über, ansonsten (mindestens ein Startkommando lief fehlerhaft) wird der Zustand auf FAILED gesetzt.

### Fehlersemantik

Wie bereits in den Anforderungen beschrieben, können Fehler bei der Durchführung von Experimenten an verschiedenen Stellen auftreten. Systemfehler treten auf, wenn die Durchführung einer Operation fehlschlägt, z.B. wenn während eines Swapin nicht genügend Ressourcen verfügbar sind. Diese Fehler führen immer zu einem Abbruch des gesamten Experimentlaufs. Das Meta-Experiment wird dabei gestoppt, d.h. alle aktiven Teilexperimente werden unterbrochen. Das Verhalten ist vom System festgelegt, und kann vom Experimentator nicht beeinflusst werden.

Eine fehlerhafte Durchführung eines Experiments, wie sie in der Erfolgsverwaltung im vorigen Abschnitt beschrieben wurde, kann zwei Auswirkungen haben. Entweder wird der Lauf unterbrochen, oder der Fehler ignoriert. Das Verhalten kann vom Benutzer durch eine Option in der Szenariobeschreibung gesteuert werden (tb-exprun-onfailure). Abhängig vom Wert der Option wird der Experimentlauf mit Durchführung aller restlichen Teilexperimente weitergeführt („ignore“) oder unterbrochen („fatal“).

### Informationsmöglichkeiten / Fortschrittskontrolle

Neben der Steuerung werden einem Experimentator Zugriff auf Informationen über den Zustand und den Fortschritt von Meta- und Teilexperimenten ermöglicht. Dazu werden die bereits in Emulab verfügbaren Möglichkeiten entsprechend angepasst und erweitert.

In der Web-Oberfläche werden bereits Informationen über Experimente mit unterschiedlichem Detaillierungsgrad gegeben. In einer Übersichtsdarstellung werden die Experimente in einer Tabelle aufgelistet, die, neben der Zuordnung zum Projekt und dem Experimentnamen, eine geschätzte Anzahl der benötigten physischen Knoten und eine kurze Beschreibung umfasst.

Zur Unterstützung von Experimentläufen wird die Darstellung angepasst und erweitert. Für Meta-Experimente wird eine zusätzliche Tabelle angezeigt, die den Status des Meta-Experiments und die Anzahl der Teilexperimente umfasst. Detaillierte Informationen über Teilexperimente werden in der Übersichtsdarstellung nicht gegeben.

Für jedes Experiment kann eine Detailansicht abgerufen werden. Die entsprechenden Webseite wird für Meta-Experimente um eine zusätzliche Liste aller Teilexperimente und deren

Zustand erweitert.

Zusätzlich zu den Informationen, die ein Experimentator vom System abfragen kann, werden in Emulab Statusinformationen nach Durchführung längerer, asynchroner Operationen durch e-Mails übermittelt. So werden nach dem Anlegen, der Aktivierung von Experimenten, nach deren Ende und dem Entfernen Log-Dateien gesendet. Für die Meta- und Teilexperimente wird diese Funktionalität übernommen und leicht angepasst. So wird z.B. nach Durchführung eines Teilexperiments der Erfolg oder Misserfolg in die Betreffzeile der e-Mail eingefügt.

## 5.3 Entwurf

Das bestehende System wird für die Unterstützung parametrisierter Experimentläufe angepasst und erweitert. In diesem Abschnitt wird beschrieben, welche Teile des Systems verändert werden. Details der Implementierung werden nicht betrachtet, und kleinere Anpassungen außer Acht gelassen.

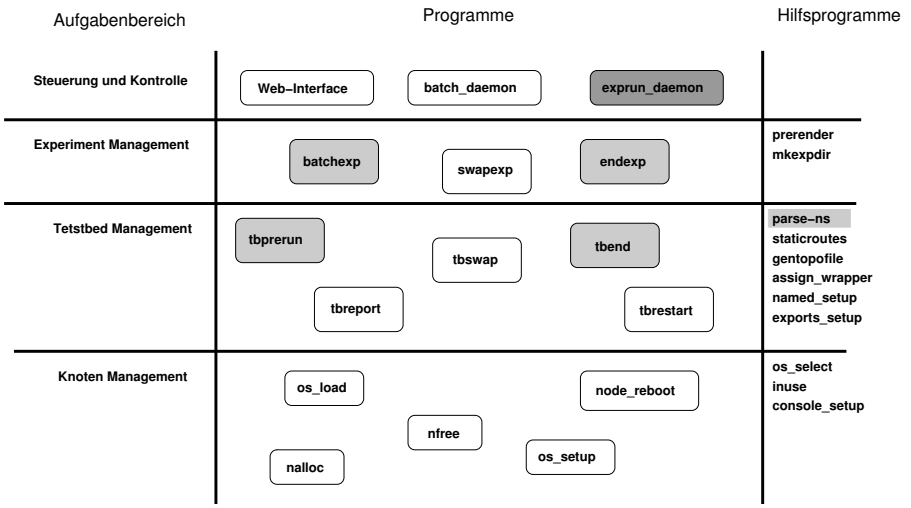



Abbildung 5.5: Erweiterte Architektur für parametrisierte Experimentläufe (vereinfacht)

In Abb. 5.5 ist nochmals die in Abschnitt 4.3 vorgestellte Architektur der Komponenten des Emulab-Systems dargestellt. Die Darstellung umfasst dabei lediglich Komponenten, die bei der Verwaltung und Durchführung von Experimenten von Bedeutung sind. Diverse kleinere Hilfsprogramme sind aus Darstellungsgründen nicht enthalten. Geänderte Komponenten sind in der Abbildung hervorgehoben dargestellt (batchexp, parse-ns, ...), eine zusätzliche, neue Komponente zur Steuerung und Koordination von Experimentläufen ist besonders gekennzeichnet (exprun_daemon).

### 5.3.1 Parser

Der Parser in Emulab wird zur Transformation einer Szenariobeschreibung von einem oTcl-Skript in ein XML-basiertes Format eingesetzt. Zur Unterstützung von parametrisierten Experimentläufen wird der Parser erweitert, so dass aus einem parametrisierten oTcl-Skript mehrere

nicht-parametrisierte oTcl-Skripte erzeugt werden können, die als Szenariobeschreibungen der Teilexperimente verwendet werden.

Der Parser stellt zur Definition einer Szenariobeschreibung Klassen bereit, die während der Interpretation des übergebenen oTcl-Skripts zur Instanziierung von miteinander verknüpften Objekten führt. Die Generierung der Ausgabe wird durch den Aufruf der Methode „run“ des Simulator-Objekts angestoßen, die für alle erstellten Objekte die Methode „updatedb“ aufruft. In „updatedb“ werden dann alle Attribute, bei einer Verbindung z.B. Name und Verlustrate, im XML-Format ausgegeben.

Zur Unterstützung einer Transformation von oTcl nach oTcl wird für alle Klassen eine weitere Methode „totclfile“ definiert. Das jeweilige Ausgabeformat wird durch einen Aufrufparameter gewählt. Um eine Transformation von oTcl nach oTcl durchzuführen, wird dann „totclfile“ verwendet, anstatt für alle Objekte die Methode „updatedb“ aufzurufen. Als Ergebnis werden oTcl-Skripte für alle Teilexperimente erzeugt. Zusätzlich wird in einer weiteren Datei die Anzahl der Teilexperimente und die Optionen des Meta-Experiments (tb-exprun-*) gespeichert.

Da eine Definition von parametrisierten Experimentläufen aus einer Anzahl von Szenariobeschreibungen entsteht, muss nach jedem Aufruf von „run“ der interne Zustand zurückgesetzt werden, da nicht bekannt ist, welche Komponenten der Parametrisierung unterzogen werden. Dazu wird die Methode „run“ erweitert, die nach der Ausgabe eines Szenarioskripts alle instanziierten Objekte löscht und globale Variablen auf ihren Ausgangswert zurücksetzt. Um eine übersichtlichere Szenariobeschreibung zu ermöglichen wird das Simulator-Objekt automatisch erneut angelegt, so dass es in einer Szenariobeschreibung nur einmal definiert werden muss.

### 5.3.2 Operationen

Die in Abschnitt 5.2.4 angesprochenen Kommandos werden von Emulab durch verschiedene Programme realisiert.

Ein Anlegen von Experimenten wird mittels batchexp und tbprerun durchgeführt. Während batchexp den gesamten Ablauf steuert, wird in tbprerun unter anderem der Parser aufgerufen. Erst nach dem Parsen ist bekannt, ob eine Szenariobeschreibung zu mehreren Experimenten führt, und damit ein Meta-Experiment und mehrere Teilexperimente erstellt werden müssen. In diesem Fall werden durch rekursiven Aufruf von batchexp in tbprerun alle Teilexperimente angelegt. Als zusätzliche Optionen werden beim rekursiven Aufruf der Name des Meta-Experiments und eine Seriennummer übergeben und in der Datenbank gespeichert, so dass eine Verknüpfung zwischen beiden Elementen entsteht.

Experimente werden durch die Programme endexp und tbend aus dem System entfernt. Ähnlich zum Anlegen führt der Aufruf von endexp zum Aufruf von tbend. Wird endexp für ein Meta-Experiment aufgerufen, werden zunächst alle Teilexperimente durch einen rekursiven Aufruf von endexp entfernt. Danach wird das Meta-Experiment entfernt.

Alle weiteren Operationen, wie Starten, Anhalten oder Pausieren, werden zur Durchführung von Experimentläufen benötigt. Sie werden in einer Experimentlauf-Steuerung realisiert, die im nächsten Abschnitt beschrieben wird.

### 5.3.3 Experimentlauf-Steuerung

Die Experimentlauf-Steuerung wird durch den Experimentlauf-Dämon (exprun_daemon) realisiert. Der Experimentlauf-Dämon arbeitet als Systemdienst im Hintergrund.

Der Experimentlauf-Dämon erfüllt zwei Aufgaben: Die Steuerung und Koordination von Meta-Experimenten, und die Steuerung und Koordination von Teilexperimenten.

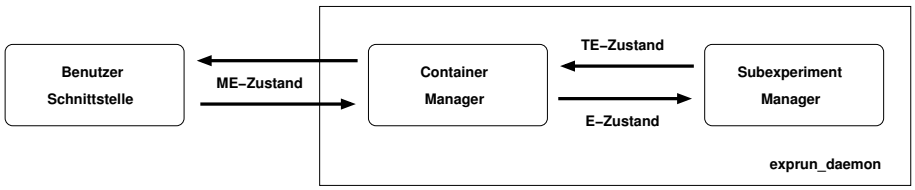



Abbildung 5.6: Komponenten der Experimentlauf-Steuerung

Infolgedessen ist der Experimentlauf-Dämon in zwei Komponenten gegliedert. Der Container-Manager (CM) stellt Funktionalitäten für Meta-Experimente bereit, der Subexperiment-Manager (SM) für Teilexperimente. In Abb. 5.6 ist das Zusammenspiel der beiden Komponenten erkennbar. Die Kommunikation erfolgt durch Ändern des Zustands von Meta- und Teilexperimenten.

Kommandos eines Benutzers werden durch Übergänge des ME-Zustands an den Container-Manager weitergegeben. Abhängig vom ME-Zustand werden dann die Teilexperimente koordiniert, indem geeignete Teilexperimente für eine Durchführung ausgewählt werden. Durchzuführende Teilexperimente werden dann dem Subexperiment-Manager übergeben. Zur Übergabe zwischen dem Container-Manager und dem Subexperiment-Manager wird der von Emulab verfügbare Zustand für Experimente (E-Zustand) verwendet

Der Subexperiment-Manager übernimmt die Steuerung der Teilexperimente. Alle Teilexperimente im Zustand QUEUED werden durchgeführt. Der Ablauf von Teilexperimenten entspricht dem von Batch-Experimenten, der bereits in Abschnitt 4.3 beschrieben wurde.

Um Festzustellen, ob ein Experiment beendet ist, wird für aktive Teilexperimente periodisch geprüft, ob alle Startkommandos der Knoten beendet wurden. In diesem Fall wird der Erfolg im TE-Zustand gespeichert, indem der Zustand auf SUCCEDED oder FAILED gesetzt wird. Damit wird dem Container-Manager Rückmeldung gegeben, welche Teilexperimente durchgeführt und beendet sind.

Die benötigte Funktionalität des Subexperiment-Managers entspricht im Wesentlichen der bereits für Batch-Experimente vom batch_daemon implementierten. Daher wurden Teile übernommen, angepasst und erweitert. Eine Erweiterung ist die nebenläufige Ausführung von Swapin- bzw. Swapout-Vorgängen. Während im batch_daemon auf die Beendigung jedes Vorgangs gewartet wird, ist dies im Experimentlauf-Dämon nicht möglich, da längere Operationen, wie z.B. ein Swapin, ansonsten zu langen Antwortzeiten führen. Für die Teilexperimente eines Meta-Experiments wird dabei zu einem Zeitpunkt nur ein Vorgang gestattet, für Teilexperimente verschiedener Meta-Experimente findet eine parallele Ausführung statt.

# Kapitel 6

# Effiziente Unterstützung von Experimentläufen

Experimente beschreiben durch ihre Szenarien Ressourcenanforderungen, für die während einer Durchführung physische Ressourcen reserviert werden müssen. Dazu werden einem Experiment Ressourcen für den Zeitraum der Durchführung zugewiesen.

Zur Steuerung des Ablaufs von Experimenten und Verwaltung von Ressourcen wird daher ein System benötigt, das durch vorab festgelegte Ziele und geeignete Algorithmen einen koordinierten Betrieb ermöglicht.

Die Effizienz kann für ein solches System aus zwei Blickrichtungen definiert werden. Eine effiziente Durchführung von Experimenten ist für einen Experimentator eine entscheidende Größe, die er im Umgang mit dem System erfährt. Die Zeit zwischen einem Kommando zur Durchführung und dem Ende eines Experimentlaufs soll aus der Sicht des Experimentators möglichst kurz sein. Eine effiziente Auslastung von Ressourcen, z.B. der physischen Knoten, ist aus der Sicht des Betreibers wünschenswert. Beides können gegensätzliche Ziele bei der Entwicklung einer Ablaufsteuerung zur effizienten Unterstützung von Experimentläufen sein.

## 6.1 Anforderungen

Experimente können aufgrund des Zusammenhangs zwischen dem Kommando zur Ausführung und dem eigentlichen Ausführungszeitpunkt unterschieden werden. Bei einer synchronen Betriebsform findet die Durchführung direkt als Folge des Kommandos zur Ausführung statt. Ist das Kommando vom Startzeitpunkt eines Experiments entkoppelt, stellt das eine asynchrone Betriebsform dar. Eine asynchrone Betriebsform wird meist durch einen Stapelbetrieb realisiert.

#### Allgemeine Ziele einer Ablaufsteuerung

Angelehnt an [Tan92] können für eine Ablaufsteuerung verschiedene Ziele angestrebt werden. Ein allgemeines Ziel ist *Fairness*. Ein System ist fair, wenn z.B. vergleichbare Experimente gleich behandelt werden. Das umfasst sowohl Startzeitpunkte als auch zur Verfügung gestellte Ressourcen. Fairness kann aber nicht nur zwischen Experimenten betrachtet werden, sondern

auch zwischen Benutzern eines Systems. Die Nutzungsdauern angebotener Dienste eines Systems müssen daher so verteilt werden, dass unter gegebenen Randbedingungen alle Benutzer einen angemessenen Anteil erhalten.

Der *Durchsatz* (throughput) beschreibt die Anzahl durchgeführter Experimente pro Zeiteinheit und stellt ein weiteres Ziel dar, das bei der Realisierung einer Ablaufsteuerung angestrebt werden kann. Ein hoher Durchsatz führt zu einer hohen Zahl an insgesamt durchgeführten Experimenten innerhalb eines Zeitraumes ohne Berücksichtigung der Durchführungsdauern einzelner Experimente.

Ein Experimentator ist allerdings oft daran interessiert, Ergebnisse innerhalb kurzer Zeit zu erhalten. Die *Rückkehrzeit* (turnaround time) gibt den Zeitraum zwischen einem Kommando zur Durchführung eines Experiments und dem Endzeitpunkt der Durchführung an. Eine minimierte Rückkehrzeit kann also ein Ziel sein, wenn die Durchführung von Experimenten für einen Experimentator in einem kurzen Zeitraum erfolgen soll.

Ressourcen sind meist nur in begrenztem Umfang vorhanden. Die *Auslastung* beschreibt das Verhältnis zwischen der Zeit, die eine Ressource belegt ist, zu einem gesamten Zeitraum. Eine hohe Auslastung kann daher als Ziel betrachtet werden, um den effizienten Einsatz von Ressourcen zu gewährleisten, z.B. um teure Ressourcen kostendeckend zu betreiben.

Der zeitlich direkte Zusammenhang zwischen einer Anfrage an ein System und deren sichtbaren Auswirkungen führen zu einem weiteren Ziel. Die *Proportionalität* beschreibt den Zusammenhang zwischen Ausführungsdauern von Aufgaben und deren Komplexität. Weicht die reale Dauer einer Aufgabe deutlich von der vom Experimentator geschätzten ab, wird das als störend empfunden. Aufgaben ähnlicher Komplexität sollten deshalb ein annähernd gleiches zeitliches Verhalten zeigen.

Je nach Einsatzgebiet und Umgebung können alle Ziele mehr oder weniger ausgeprägt in die Entwicklung einer Ablaufsteuerung einfließen. Zur Umsetzung der Ziele in einer Ablaufsteuerung muss der Zeitpunkt der Durchführung, die Auswahl von Experimenten und deren Reihenfolge vorab geplant und festgelegt werden.

### Dienste einer Ablaufsteuerung

Die Ablaufsteuerung eines Systems kann einem Benutzer verschiedene Dienste bereitstellten. Diese Dienste ermöglichen es, dem Benutzer Einfluss auf den Ablauf eines Experiments, wie z.B dessen Startzeitpunkt, zu nehmen.

Der Zeitpunkt der Durchführung eines Experiments kann durch drei Werte charakterisiert werden: dem frühesten Startzeitpunkt, der Dauer und dem spätesten Endzeitpunkt. Je nach Festlegung der Werte sind verschiedene Ausführungsmodelle möglich, die eine Durchführung eines Experiments in einem bestimmten Zeitintervall ermöglichen. Die Festlegung des Startzeitpunktes bestimmt den frühesten Termin zur Durchführung. Wird die aktuelle Zeit angegeben, so wird ein Experiment zum nächstmöglichen Zeitpunkt durchgeführt.

Wird die Dauer und der Endzeitpunkt eines Experiments festgelegt, wird damit der späteste Startzeitpunkt bestimmt. Der reale Startzeitpunkt der eigentlichen Durchführung wird dabei dem System überlassen. Gleiches gilt für die Angabe aller drei Werte, die dann durch den Start- und Endzeitpunkt ein Zeitintervall festlegen, ohne den Zeitpunkt der Durchführung direkt zu

bestimmen. Die Umsetzung der angegebenen Werte kann dabei als ein gewünschtes Zeitintervall verstanden werden oder als Garantie vom System durchzusetzen sein.

Eine weitere Möglichkeit stellen Modelle dar, die den Ablauf nicht über die Angabe von Zeitpunkten beeinflussen, sondern anhand von Relationen zwischen Experimenten Einfluss auf die Ausführungsreihenfolge nehmen. Ein Prioritätssystem ist z.B. eine Möglichkeit, Experimente in Klassen gleicher Dringlichkeit einzuordnen. Experimente mit hoher Priorität werden vom System bevorzugt durchgeführt, während die Durchführung von Experimenten mit geringer Priorität auf einen späteren Zeitpunkt verschoben werden kann.

### Planung des Ablaufs

Der Ablauf muss, unabhängig von den Möglichkeiten eines Benutzers das System zu beeinflussen, geplant werden. Ziel der planerischen Tätigkeiten ist es, eine Ausführungsreihenfolge für Experimente festzulegen.

Eine Planung kann verschiedene Zeiträume umfassen. Eine zeitpunktorientierte Planung bestimmt aus dem aktuellen Zustand des Systems und den zur Ausführung anstehenden Experimenten jeweils nur das nächste. Eine zeitraumorientierte Planung bestimmt die Ausführungsreihenfolge mehrerer Experimente. Die Planung erfolgt dabei vorausschauend über einen längeren Zeitraum, und kann, durch geeignete Auswahl und Zusammenstellung mehrerer Experimente, einen höheren Grad der Optimierung des gesetzten Ziels (Fairness, Auslastung, ...) erreichen.

Eine Ablaufsteuerung benötigt für eine planerische Tätigkeit Informationen, die als Grundlage für Entscheidungen über den weiteren Ablauf dienen. Die Menge und Qualität der Informationen ist dabei entscheidend für die Erreichung der gesetzten Ziele. Die Planung umfasst drei Aufgabenbereiche:

- Feststellen benötigter Ressourcen,
- Finden freier Ressourcen mit passenden Eigenschaften und
- Auswählen und Belegen von Ressourcen für die Dauer eines Experiments.

Aus den Szenariobeschreibungen ergeben sich die Ressourcenanforderungen. Die Anforderungen umfassen dabei nicht nur die Anzahl der Knoten oder Verbindungen und deren Eigenschaften, sondern auch weitere Anforderungen, wie z.B. eine spezielle Hardware-Ausstattung. Diese Anforderungen sind bei einer Planung zu berücksichtigen.

Die Anforderungen müssen von den physischen Ressourcen erfüllt werden. Dazu müssen physische Ressourcen, deren Eigenschaften und deren Belegungszustand bekannt sein. Ist eine mögliche Belegung gefunden, die alle gewünschten Eigenschaften erfüllt, können die Ressourcen einem Experiment für die Dauer der Durchführung zugewiesen werden.

Neben der Auswahl der Ressourcen ist vor allem deren Belegungsdauer wichtig. Sie bestimmt sich aus der Laufzeit eines Experiments. Die Laufzeit kann, sofern sie nicht aus einer Szenariobeschreibung hervorgeht oder vom Experimentator festgelegt wird, zum Beispiel aus bereits durchgeführten Läufen des gleichen Experiments geschätzt werden und unterstützend in die Planung einbezogen werden.

**Effiziente Gestaltung von Experimentläufen**

Experimentläufe bestehen aus einer Menge von Teilexperimenten. Die dabei zusätzlich vorhandenen Informationen können für eine Planung zur effizienten und optimalen Ressourcenauslastung und Bestimmung einer Ablaufreihenfolge verwendet werden.

Es ergeben sich zwei Möglichkeiten für eine effiziente Unterstützung von Experimentläufen:

- Effiziente Gestaltung des *gesamten* Experimentlaufs und
- Effiziente Gestaltung *einzelner Teilexperimente* eines Experimentlaufs.

Eine effiziente Gestaltung eines Experimentlaufs kann z.B. die Durchsatzoptimierung als Ziel haben. Soll ein Lauf möglichst schnell beendet werden, so können unabhängige Experimente parallel zueinander ausgeführt werden (siehe Abschnitt 5.1). Eine parallele Durchführung mehrerer Experimente bedingt allerdings auch eine Belegung von entsprechend mehr Ressourcen, die von einer Ablaufsteuerung begrenzt werden muss.

Für eine effiziente Gestaltung einzelner Teilexperimente bei der Durchführung eines Experimentlaufs können weitere Eigenschaften ausgenutzt werden. In einem parametrisierten Lauf sind die Änderungen in den Szenarien einzelner Teilexperimente zueinander oft nur gering, da eine Evaluation meist durch sich inkrementell ändernde Parameter erfolgt. So kann es sein, dass nur Aufrufparameter eines zu evaluierenden Programms geändert werden oder sich lediglich die Anzahl der beteiligten Knoten ändert. Die Nutzung dieses vorab vorhandenen Wissens kann zu effizienteren Zyklen bei Ablauf einzelner Teilexperimente führen.

## 6.2 Spezifikation

Im letzten Kapitel wurden die Komponenten diskutiert, die für eine Unterstützung von parametrisierten Experimentläufen benötigt werden. Dabei wurden Aufgaben, die von einer Ablaufsteuerung für eine effiziente Durchführung von Experimentläufen zu erfüllen sind, jedoch zunächst nicht betrachtet.

### 6.2.1 Stand in Emulab

Durch die von Emulab unterstützten Betriebsarten können zwei Arten von Experimenten unterschieden werden. Interaktive Experimente, die manuell durch einen Experimentator gesteuert werden und Batch-Experimente, bei denen ein Experimentator ein Kommando zur Durchführung absetzt und die eigentliche Durchführung automatisiert stattfindet. Beide Arten von Experimenten benötigen Ressourcen.

Die Ressourcenauswahl und -reservierung wird in Emulab direkt während des Umsetzens einer Szenariobeschreibung auf die physische Umgebung vorgenommen. Es wird also lediglich zum Zeitpunkt einer Aktivierung die gegenwärtige Ressourcenverfügbarkeit betrachtet. Sind die Anforderungen einer Szenariobeschreibung erfüllbar, wird ein Experiment umgehend aktiviert. Schlägt die Ressourcenreservierung fehl, wird der Swapin bei interaktiven Experimenten abgebrochen und bei Batch-Experimenten der Versuch zusätzlich nach jeweils 15 Minuten

wiederholt. Der Ablauf wird dabei nicht durch eine globale Ressourcenverwaltung unterstützt, sondern findet für mehrere Experimente unabhängig voneinander statt.

Daher kann auch keine globale Planung zur Durchführung von Experimenten bereitgestellt werden. Eine vorwärtsorientierte Planung ist nicht möglich, und es können keine Garantien über Zeitpunkte für zukünftig auszuführende Experimente gegeben werden. Eventuelle zusätzliche Informationen, z.B. die Dauer eines Experiments aus einer früheren Durchführung, werden daher nicht betrachtet. Durch Angabe einer maximalen Dauer kann ein Experiment zwar zeitlich begrenzt sein, die Angabe wird aber beim Anlegen eines Experiments nicht erzwungen und kann daher nicht als Grundlage einer zukünftigen Planung dienen. Obwohl keine Planung erfolgt, wäre es möglich, die Durchführung von mehreren Experimenten verschiedener Experimentatoren einzuschränken und damit zwischen den Benutzern des Systems ein faires Verhalten zu erzielen. Eine derartige Koordination, z.B. durch eine Zuordnung von Prioritäten, wird jedoch nicht unterstützt. Insgesamt wird damit ein Verhalten erzeugt, das einem FCFS-Schema (first come first serve) entspricht.

Da keine globale Ressourcenverwaltung existiert und der Ablauf nicht geplant wird, sind Vorabreservierungen von Ressourcen zur Durchführung von Experimenten zu einem bestimmten Zeitpunkt oder in einem zukünftigen Zeitintervall nicht möglich. Weiterhin können Experimente nicht unterbrochen und zu einem späteren Zeitpunkt fortgesetzt werden. Eine vorzeitige Beendigung eines Experiments durch eine Ablaufsteuerung ist zwar möglich, ein Experiment kann aber anschließend nicht reaktiviert und nahtlos fortgesetzt werden. Ein Experiment ist damit bei einer Unterbrechung gescheitert und muss in einem erneuten Versuch vollständig wiederholt werden.

Aus den Betrachtungen wird deutlich, welches Ziel damit erreicht werden kann: die Durchsatzmaximierung. Es werden zu einem Zeitpunkt möglichst viele Experimente durchgeführt. Durch die isolierte Betrachtung von Experimenten und eine fehlende Planungskomponente kann das System aber keine Fairness garantieren. Eine Koordination findet daher unabhängig vom Emulab-System durch kooperative Zusammenarbeit der Benutzer und manuelle Steuerung durch die Administratoren statt.

### 6.2.2 Ansatz

Zusätzlich zu interaktiven Experimenten und Batch-Experimenten sind mit der Unterstützung von parametrisierbaren Experimentläufen ebenfalls Meta-Experimente und deren Teilexperimente in eine Betrachtung mit einzubeziehen. Zur Koordination und Steuerung von Experimentläufen wird zunächst die Eingliederung in das bestehende System betrachtet und danach auf eine effiziente Unterstützung für parametrisierte Experimentläufe eingegangen. Dabei sind folgende Beziehungen von Bedeutung:

- Koordination zwischen Batch- bzw. interaktiven Experimenten und Meta-Experimenten,
- Koordination zwischen mehreren Meta-Experimenten, und
- Koordination zwischen Teilexperimenten eines Meta-Experiments.

Da Meta-Experimente ganzheitlich mit allen Teilexperimenten ausgeführt werden, müssen sie derart in das System eingebunden werden, dass sich gegenüber den anderen Experimenten ein faires Verhalten ergibt. Da in Emulab keine Planung erfolgt und lediglich aufgrund verfügbarer Ressourcen über eine Aktivierung entschieden wird, kann die „Koordination“ wie folgt erreicht werden: Sind für ein Meta-Experiment genügend Ressourcen vorhanden, wird es durchgeführt. Die Antwort auf die Frage, was „genügend Ressourcen“ sind, wird auf später verschoben.

Die Koordination zwischen Meta-Experimenten wird ebenfalls geregelt, indem eine Aktivierung erfolgt, sobald genügend Ressourcen verfügbar sind. Zusätzlich wird die Koordination zwischen Meta-Experimenten jedoch durch ein einfaches Prioritätenschema verfeinert. Werden mehrere Meta-Experimente gleichzeitig aktiviert, bestimmen deren Prioritäten die Ausführungsreihenfolge. Die Priorität ist jedoch nur von Bedeutung, wenn nicht genügend Ressourcen vorhanden sind, um alle Meta-Experimente gleichzeitig durchführen zu können. Eine höhere Priorität führt dabei zu bevorzugter Durchführung eines Meta-Experiments. Die Priorität wird vom Experimentator in der Szenariobeschreibung definiert und durch fünf Abstufungen realisiert (tb-exprun-priority; siehe Abschnitt 5.2.3). Da die Angabe des Wertes keiner Einschränkung unterliegt, findet die Koordination allerdings weiterhin kooperativ zwischen den Benutzern statt.

Für die Koordination zwischen Teilexperimenten eines Meta-Experiments wurden bereits in Abschnitt 6.1 zwei Möglichkeiten angedeutet, indem zwischen einer effizienten Gestaltung des gesamten Experimentlaufs und einer effizienten Gestaltung einzelner Teilexperimente eines Experimentlaufs unterschieden wurde. Beide Möglichkeiten werden kombiniert eingesetzt. Ziel ist es, die gesamte Durchführungsdauer eines Meta-Experiments zu minimieren, um damit eine Durchsatzmaximierung zu erreichen.

Eine effiziente Gestaltung des gesamten Experimentlaufs wird durch eine parallele Durchführung der Teilexperimente eines Meta-Experiments erreicht. Dabei ist zu beachten, dass eine gleichzeitige Durchführung mit einer erhöhten Ressourcenauslastung verbunden ist. Der Parallelisierungsgrad muss also auf geeignete Weise eingeschränkt werden, um ein unfaires Verhalten des Systems gegenüber anderen Benutzern durch eine gleichzeitige Durchführung vieler Teilexperimente zu verhindern.

Eine effiziente Gestaltung einzelner Teilexperimente eines Experimentlaufs wird durch einen speziellen Mechanismus erreicht, der eine schnelle Wiederverwendung von Knoten in einem neuen Kontext ermöglicht. Dabei werden physische Knoten nach Beendigung eines Teilexperiments nicht dem System übergeben, sondern für ein folgendes Teilexperiment zunächst weiterhin reserviert gehalten und deren Konfiguration zur Durchführung des folgenden Teilexperiments angepasst. Dadurch ist es möglich die Dauer für die Bereitstellung der Ausführungsumgebung in einem Experimentlauf zu reduzieren. Beide Möglichkeiten, die parallelisierte Durchführung und die Wiederverwendung von Knoten, werden kombiniert eingesetzt und im Folgenden näher betrachtet.

### 6.2.3 Effiziente Unterstützung von Experimentläufen

Die gesamte Dauer eines Meta-Experiments hängt von der Zahl der Teilexperimente und deren Durchführungsdauern ab. Die Durchführungsdauer eines Meta-Experiments kann daher deut-

lich verkürzt werden, wenn Teilexperimente parallel zueinander ausgeführt werden.

Eine gleichzeitige Durchführung von Teilexperimenten ist möglich, wenn sie voneinander unabhängig sind (siehe Abschnitt 5.1). Bestehen zwischen allen Teilexperimenten eines Meta-Experiments keine Abhängigkeiten können daher theoretisch alle Teilexperimente gleichzeitig durchgeführt werden. Da im Zuge der Diplomarbeit keine Abhängigkeiten unterstützt werden, ist eine parallele Durchführung aller Teilexperimente immer möglich. Dabei entstehende Probleme durch eine hohe Ressourcenauslastung müssen durch eine Beschränkung der Parallelität abgefangen werden.

### Beschränkung der Parallelität

Für Teilexperimente werden zur Durchführung Ressourcen benötig und reserviert. Ressourcen stehen allerdings nur in einer begrenzten Menge zur Verfügung. Der real zu erreichende Parallelisierungsgrad hängt demnach auch von den verfügbaren Ressourcen ab. Es können daher nicht beliebig viele Teilexperimente gleichzeitig durchgeführt werden.

Weiterhin ist eine Einschränkung der Parallelität auch Voraussetzung, um ein faires Verteilen der Ressourcen auch zwischen Meta-Experimenten und Batch- bzw. interaktiven Experimenten sicherzustellen. Würde ein Experimentlauf alle durchführbaren Teilexperimente zugleich starten und dabei einen Großteil der Ressourcen belegen, wäre der interaktive Betrieb und Stapelbetrieb aufgrund fehlender Ressourcen nicht mehr möglich. Daher muss die maximale Parallelität, die theoretisch aufgrund der Abhängigkeiten zu erreichen wäre, eingeschränkt werden.

Eine Beschränkung kann auf zwei Arten erfolgen. Entweder wird eine maximale Anzahl gleichzeitiger Teilexperimente für ein Meta-Experiment festgelegt, oder die maximale Anzahl reservierbarer Knoten für alle Teilexperimente eines Meta-Experiments begrenzt. Die maximale Anzahl gleichzeitiger Teilexperimente scheint zunächst intuitiver zu sein. Nur aufgrund der Anzahl der Teilexperimente kann die reale Ressourcenauslastung aber nicht beschrieben werden. Eine bessere Alternative stellt daher die Beschränkung der Anzahl der Knoten dar, die bei der Durchführung eines Meta-Experiments maximal belegt werden dürfen.

### Verfahren zur Parallelisierung

Eine geeignete Parallelisierung bei gleichzeitiger Beschränkung der Ressourcen wird durch die Limitierung der Knotenanzahl erreicht, die ein Meta-Experiment bei Durchführung der Teilexperimente belegen darf. Die Zahl wird bestimmt durch das Maximum der Anzahl der Knoten eines Teilexperiments über alle Teilexperimente eines Meta-Experiments. Das „größte“ Teilexperiment bestimmt also die Zahl der Knoten, die während der Durchführung eines Meta-Experiments belegt werden dürfen.

Um diese Anzahl nicht zu überschreiten, werden die Knoten nur einmal bei Durchführung des größten Teilexperiments belegt. Danach werden sie nicht wieder als freie Ressourcen dem System übergeben, sondern während der gesamten Durchführung entweder dem Meta-Experiment oder Teilexperimenten zugeordnet.

Der Ablauf eines Meta-Experiments ist dann wie folgt zu beschreiben. Ein Meta-Experiment wird zunächst durch ein Kommando des Benutzers in eine Warteschlange gestellt. Peri-

odisch wird geprüft, ob genügend freie Knoten für die Durchführung des größten Teilexperiments verfügbar sind. Danach wird zunächst das größte Teilexperiment durchgeführt. Die inital Reservierung der Knoten zur Durchführung des Meta-Experiments findet also während der Durchführung des ersten, größten Teilexperiments statt. Nach dessen Durchführung werden die reservierten Knoten nicht wieder dem System übergeben, sondern umgehend dem Meta-Experiment zugeordnet. Die Knoten können dann zur Durchführung der weiteren, restlichen Teilexperimente verwendet werden.

Dazu werden alle restlichen Teilexperimente geprüft, ob die Anzahl der im Meta-Experiment gehaltenen Knoten für eine Durchführung weiterer Teilexperimente ausreichend ist. Die Teilexperimente werden dabei in absteigender Reihenfolge der Knotenanzahl geprüft. Da die Anzahl der benötigten Knoten aller verbliebenen Teilexperimente kleiner oder gleich zu der des vorigen Teilexperiments ist, wird die Bedingung mindestens einmal erfüllt. Dem gewählten Teilexperiment werden dann vor der Aktivierung die benötigten Knoten aus dem Meta-Experiment zugeordnet. Für weitere Teilexperimente wird iterativ fortgefahren und geprüft, ob die restlichen Knoten zur Durchführung weiterer Teilexperimente ausreichend sind. Die Teilexperimente werden dann ebenfalls aktiviert und parallel ausgeführt.

Der Ansatz führt also dazu, dass mehrere Teilexperimente parallel laufen können, dabei die Ressourcenauslastung auf die Anzahl der Knoten des größten Teilexperiments beschränkt wird, und damit die Parallelität eingeschränkt wird. Ein Meta-Experiment mit Teilexperimenten, die alle eine gleiche Anzahl von Knoten benötigen, wird also serialisiert durchgeführt. Ein Meta-Experiment dessen größtes Teilexperiment zehn Knoten benötigt und dessen weitere Teilexperimente nur zwei Knoten benötigen, führt zunächst das Teilexperiment mit zehn Knoten aus und danach jeweils fünf der restlichen Teilexperimente parallel. Es kann angenommen werden, dass oftmals Experimentläufe mit einer steigenden Anzahl von Knoten durchgeführt werden. Dabei entstehen Teilexperimente mit Größen von einem Knoten bis zu einer maximalen Anzahl von Knoten. In diesem Fall werden nach Durchführung des größten Teilexperiments immer zwei Teilexperimente gleichzeitig durchgeführt: das größte und das kleinste verbliebene Teilexperiment.

Die einmalige Reservierung der Knoten bei Durchführung des größten Teilexperiments stellt zudem sicher, dass nach einer erfolgreichen Aktivierung und damit einhergehender Reservierung der Knoten für alle weiteren Teilexperimente ebenfalls genügend Ressourcen zur Durchführung vorhanden sind. Die Aktivierung der folgenden Teilexperimente kann damit nicht mehr aufgrund fehlender Ressourcen fehlschlagen und ein Meta-Experiment kann nach erfolgreicher Aktivierung des ersten Teilexperiments komplett durchgeführt werden.

Das Verfahren ist weiterhin auch bei Unterbrechen und Wiederaufnahme eines Meta-Experiments anwendbar. Die Zahl der für ein Meta-Experiment reservierten Knoten wird dann durch die verbliebenen Teilexperimente bestimmt.

#### Beispiel: Internet Szenario

Zur Verdeutlichung des Verfahrens zur Parallelisierung wird der Ablauf bei Durchführung des in Abschnitt 5.2.3 parametrisierten Internet Szenarios betrachtet. Der Experimentlauf wird durch ein Meta-Experiment gebildet, das durch Parametrisierung neun Teilexperimente umfasst.

| Teilexperiment | Name | Anzahl Knoten |
|---|---|---|
| Teilexperiment 1 | iNET-ER-1 | 4 |
| Teilexperiment 2 | iNET-ER-2 | 4 |
| Teilexperiment 3 | iNET-ER-3 | 4 |
| Teilexperiment 4 | iNET-ER-4 | 8 |
| Teilexperiment 5 | iNET-ER-5 | 8 |
| Teilexperiment 6 | iNET-ER-6 | 8 |
| Teilexperiment 7 | iNET-ER-7 | 12 |
| Teilexperiment 8 | iNET-ER-8 | 12 |
| Teilexperiment 9 | iNET-ER-9 | 12 |

Tabelle 6.1: Benötigte Knoten der Teilexperimente des Internet Szenarios

Das Szenario wird durch zwei Parameter variiert. Die Topologie eines Teilexperiments wird durch die Anzahl der an die beiden Router angeschlossenen Knoten verändert. Da für eine Topologie dabei die Bandbreite des Flaschenhalses ebenfalls durch drei Werte variiert wird, besitzen jeweils drei Teilexperimente die gleiche Topologie und damit eine gleiche Anzahl von Knoten. In Tab. 6.1 sind für alle Teilexperimente deren Namen und die zur Durchführung benötigten Knoten aufgeführt.

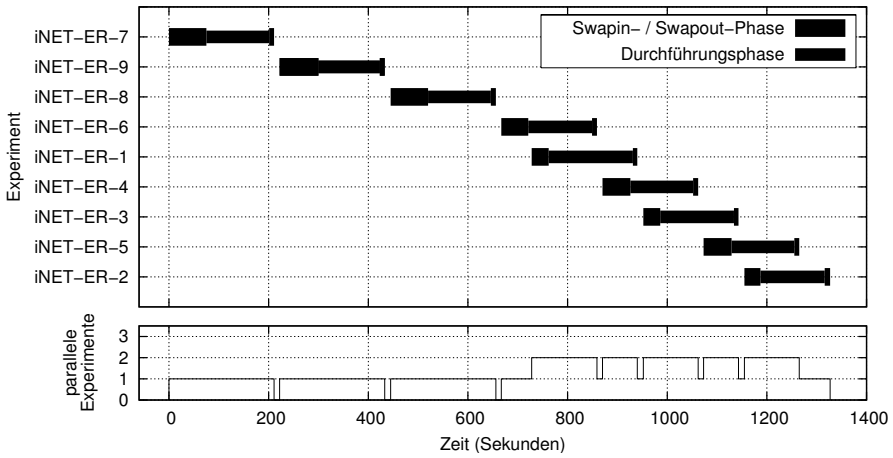



Abbildung 6.1: Ablauf der Teilexperimente des Internet Szenarios

Die Abfolge der Teilexperimente während der Durchführung des Meta-Experiments ist in Abb. 6.1 dargestellt. Die Darstellung ist in zwei Bereiche gegliedert. Im oberen Teilabschnitt ist die Abfolge aller Teilexperimente dargestellt. Zusätzlich sind die Swapin-, Durchführungs- und Swapout-Phase der Teilexperimente durch verschiedene Strichstärken zu erkennen. Der untere Teilabschnitt stellt die Anzahl der gleichzeitig aktiven Teilexperimente dar.

Der Ablauf ist dann wie folgt. Zunächst wird ein (beliebiges) Teilexperiment mit der maximalen Knotenzahl durchgeführt (iNET-ER-7). Dem Meta-Experiment stehen danach 12 Knoten zur Durchführung der restlichen Teilexperimente zu Verfügung. Nacheinander werden dann

die beiden anderen Teilexperimente, die 12 Knoten benötigen, durchgeführt (iNET-ER-9, iNET-ER-8).

Danach werden die zur Verfügung stehenden Knoten zur gleichzeitigen Durchführung mehrerer Teilexperimente verwendet und die Knoten zwischen ihnen aufgeteilt. Dazu wird zuerst ein Teilexperiment mit acht Knoten gewählt (iNET-ER-6), und die verbliebenen vier Knoten des Meta-Experiments zur parallelen Durchführung eines zweiten Teilexperiments verwendet (iNET-ER-1). Analog wird dann mit der Durchführung von jeweils einem Teilexperiment mit acht Knoten und parallel dazu einem Teilexperiment mit vier Knoten fortgefahren. Alle 12 Knoten sind dabei während der gesamten Durchführung entweder aktiven Teilexperimenten oder dem Meta-Experiment zugeordnet, und werden erst nach Beendigung des letzten Teilexperiments freigegeben.

Da die Swapin- und Swapout-Vorgänge eines Meta-Experiments aus Gründen der Lastverteilung serialisiert werden, findet die Aktivierung der Teilexperimente bei paralleler Durchführung leicht verzögert statt. Zwei parallel durchzuführende Teilexperimente werden nacheinander aktiviert, die eigentliche Durchführungsphase ist dann aber parallel.

### 6.2.4 Effiziente Unterstützung von Teilexperimenten

Im vorigen Abschnitt wurde die Parallelisierung betrachtet, die eine Verkürzung der gesamten Dauer durch die gleichzeitige Durchführung von Teilexperimenten erreicht. Alle reservierten Knoten sind dabei während der gesamten Durchführungsdauer entweder dem Meta-Experiment oder Teilexperimenten zugeordnet.

Die einmalige Reservierung von Knoten für einen Experimentlauf wird aber noch auf eine weitere Weise ausgenutzt, um den Zeitraum zwischen zwei Teilexperimenten zu verkürzen. Ein Experiment wird durch den Swapin-Vorgang, die eigentliche Durchführung und den Swapout-Vorgang in drei Phasen eingeteilt. Die Länge der Durchführungsphase wird vom Experimentator gesteuert und kann nicht beeinflusst werden. Die Swapout-Phase ist nahezu konstant und von eher kurzer Dauer. Die Zeit zur Bereitstellung einer Ausführungsumgebung zwischen zwei Teilexperimenten kann aber verkürzt werden.

Analysen haben ergeben, dass während der Swapin-Phase der größte Aufwand durch die Bereitstellung und Konfiguration der Knoten verursacht wird [WLS+02]. Die Dauer für das Neustarten der Knoten und Aufspielen von Festplattenabbildern nehmen dabei den größten Anteil ein. Eine Verkürzung der Swapin-Phase für ein Experiment kann daher erreicht werden, indem Neustarts und Aufspielen von Festplattenabbildern vermieden werden. Dazu werden bereits reservierte Knoten aus vorigen Teilexperimenten wiederverwendet.

Das Aufspielen von Festplattenabbildern auf physische Knoten wird in Emulab aus zwei Gründen durchgeführt. Zum einen wird vor jedem Experiment ein definierter Zustand des physischen Knotens erreicht. Zum anderen werden Festplattenabbilder immer neu aufgespielt, um Zugriffe auf Daten auf physischen Knoten eines durchgeführten Experiments dem Experimentator eines folgenden Experiments nicht zugänglich zu machen. Für parametrisierte Experimentläufe ist eine Wiederverwendung von Knoten zwischen Teilexperimenten aber ohne diese Sicherheitsbedenken möglich, da die Teilexperimente eines Experimentlaufs immer nur von einem Experimentator durchgeführt werden.

### Wiederverwendung von Knoten

Während der Durchführung eines Experimentlaufs werden physische Knoten durch das Verfahren der Parallelisierung nur einmal reserviert. Nach Beenden eines Teilexperiments werden die Knoten so belassen wie sie sind und lediglich dem Meta-Experiment zugeordnet. Ein Neustarten ist nicht zwingend notwendig, ebenso müssen Festplattenabbilder nur erneut aufgespielt werden, wenn eine Parametrisierung zum Einsatz verschiedener Betriebssysteme führt.

Dennoch müssen Knoten bei Reservierung für ein folgendes Teilexperiment neu konfiguriert werden. Zur Rekonfiguration eines physischen Knotens in einem Folgeexperiment, wird ein bereits existierender Mechanismus von Emulab ausgenutzt. Der Mechanismus wird ursprünglich verwendet, um aktive Experimente modifizieren zu können. Nach einer Modifikation eines aktiven Experiments werden unter anderem alle Knoten „rekonfiguriert“. Die Rekonfiguration beinhaltet alle Aktionen, die zur Bereitstellung der Ausführungsumgebung auf den Knoten vorgenommen werden müssen. Dazu gehören z.B. das Anpassen von IP-Adressen, des Rechnernamens oder der Start des Ereignissystems. Das Rekonfigurieren wird dann für parametrisierte Experimentläufe ausgenutzt, indem gehaltene Knoten für ein Folgeexperiment lediglich rekonfiguriert werden.

Um eine größere Flexibilität zu erreichen, werden einem Experimentator jedoch neben der Rekonfiguration weitere Optionen für eine Wiederverwendung gegeben. In der Szenariobeschreibung kann dazu ein Befehl definiert werden, der das Verhalten steuert (tb-exprunonreclaim). Dabei werden folgende Arten der Wiederverwendung ermöglicht:

- Unmittelbare Rekonfiguration der Knoten (reconfigure),
- Neustarten der Knoten mit anschließender Konfiguration (reboot), oder
- erneutes Aufspielen des Betriebssystems (reload).

Als Standardverhalten ist eine Rekonfiguration festgelegt. Der bei Wiederverwendung entstehende Aufwand und die damit einhergehende Zeitdauer ist dabei von der gewählten Art der Wiederverwendung abhängig. Eine Wiederverwendung mittels Rekonfiguration benötigt nur wenig Zeit, während ein Neustart und vor allem ein erzwungenes Aufspielen von Festplattenabbildern länger benötigen.

### Grenzen bei Wiederverwendung von Knoten

Eine Wiederverwendung von Knoten durch Rekonfiguration oder Neustarten kann allerdings zu unerwünschten Effekten führen. Nach Durchführung eines Teilexperiments wird ein Knoten so belassen wie er ist. Eventuelle Veränderungen am Dateisystem bleiben bestehen und von einem Experimentator gestartete Dienste werden nicht automatisch beendet.

Diese Änderungen können nicht erkannt werden. Ein Löschen von wichtigen Systemdateien führt also dazu, dass ein folgendes Teilexperiment unter Umständen nicht durchgeführt werden kann. Solche Veränderungen nach Beendigung oder vor dem Start von Teilexperimenten zu handhaben, bleibt Aufgabe des Experimentators.

Alle parametrisierbaren Komponenten eines Szenarios werden jedoch während der Rekonfiguration und Bereitstellung der Ausführungsumgebung korrekt umgesetzt. Dazu gehören die

Konfiguration von Netzschnittstellen und Rechnernamen, ebenso wie ein Neustart des Ereignissystems. Die automatische Installation von Software ist auch bei Durchführung von Experimentläufen mittels Rekonfiguration oder Neustart möglich.

## 6.3 Entwurf

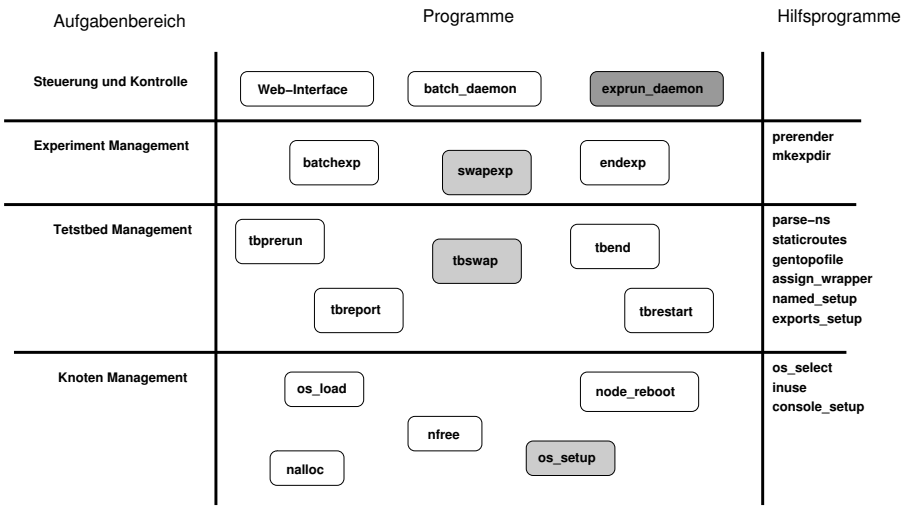



Abbildung 6.2: Erweiterte Architektur für effiziente Experimentläufe (vereinfacht)

In Abb. 6.2 ist nochmals die in Abschnitt 4.3 vorgestellte Architektur der Komponenten des Emulab-Systems dargestellt. Diverse kleinere Hilfsprogramme sind aus Darstellungsgründen nicht enthalten. Für die effiziente Unterstützung von Experimentläufen sind geänderte Komponenten in der Abbildung hervorgehoben dargestellt.

Die Parallelisierung wird durch die Experimentlauf-Steuerung (exprun_daemon) koordiniert. Eine Wiederverwendung und Rekonfiguration wird durch Anpassungen bereits existierender Komponenten ermöglicht (tbswap, os_setup, ...).

### 6.3.1 Parallelisierung

Die Durchführung von Experimentläufen wird von der Experimentlauf-Steuerung koordiniert und gesteuert (exprun_daemon). Die Steuerung und Koordination der Meta-Experimente wird dabei vom Container-Manager vorgenommen, während der Subexperiment-Manager die Durchführung von Teilexperimenten übernimmt.

Die Steuerung der Parallelisierung ist daher Aufgabe des Container-Managers, der entscheidet wann und welche Teilexperimente durchgeführt werden. Zur Parallelisierung und deren Einschränkung über eine maximale Anzahl von Knoten, werden Knoten einmalig vom ersten Teilexperiment reserviert und dann durch Zuordnung vom Meta-Experiment zu den Teilexperimenten und umgekehrt über die gesamte Durchführungsdauer gehalten.

Dazu wird ein existierender, spezieller Mechanismus verwendet, der es erlaubt Knoten umgehend und ohne weitere Modifikation von einem Experiment einem anderen zuzuordnen. Erreicht werden kann das Verhalten für einen Knoten durch einen Eintrag in die Tabelle next_reserve der Datenbank. Ein Freigeben eines Knotens (nfree) führt normalerweise dazu, dass er an das System zurückgegeben wird. Besteht ein Eintrag in der Tabelle next_reserve und wird ein Knoten dann „freigegeben“, wird er jedoch umgehend dem Experiment zugeordnet, das durch den next_reserve-Eintrag bestimmt wird.

Die Durchführung eines Meta-Experiments wird angestoßen, indem ein Kommando eines Benutzers zum Übergang des ME-Zustands in den Zustand QUEUED führt. Es wird dann periodisch geprüft, ob genügend freie Knoten für die Durchführung des größten Teilexperiments verfügbar sind. Die physischen Knoten werden dabei erst während der Swapin-Phase des ersten Teilexperiments belegt.

Nach Beendigung des ersten Teilexperiments werden die reservierten Knoten dann dem Meta-Experiment zugeordnet. Die Zuordnung der Knoten zum Meta-Experiment nach Beendigung eines Experiments muss vor der Freigabe der Knoten in der Swapout-Phase erfolgen. Daher werden an entsprechender Stelle (tbswap) direkt vor der Freigabe next_reserve-Einträge für alle reservierten Knoten erstellt. Vor Aktivierung eines folgenden Teilexperiments werden die benötigten Knoten des Teilexperiments bestimmt, und im Container-Manager ebenfalls durch einen next_reserve-Eintrag bereitgestellt. Für alle weiteren Teilexperimente wird analog fortgefahren.

Nach Durchführung des gesamten Meta-Experiments werden dann alle reservierten Knoten ohne Anlegen eines next_reserve-Eintrags vom Container-Manager freigegeben und sind damit für andere Experimente wieder verfügbar.

### 6.3.2 Wiederverwenden von Knoten

Die vom Meta-Experiment einem Teilexperiment zugeordneten Knoten werden während der Swapin-Phase des Teilexperiments aufgrund der in der Szenariobeschreibung gewählten Option für eine Wiederverwendung in ihrem Verhalten beeinflusst.

Emulab verwaltet für jeden Knoten eine Zustandsmaschine, die Auskunft über den Reservierungszustand und weitere damit verbundene Eigenschaften beschreibt. Während der Aktivierung eines Experiments steuert diese Zustandmaschine unter anderem die für einen Knoten durchzuführende Aktion. Während der Swapin-Phase nimmt ein Programm (os_setup) dann abhängig vom Zustand entsprechende Aktionen vor.

Durch Setzen des Zustands während der Swapin-Phase kann die Aktion, die bei einer Wiederverwendung erfolgen soll, angepasst werden (tbswap). Für eine Rekonfiguration wird der Zustand auf RES_RECONFIG gesetzt, um einen Knoten neu zu starten und anschließend zu konfigurieren, wird der Zustand auf RES_INIT_DIRTY gesetzt.

Das Aufspielen eines Festplattenabbildes auf die Knoten vor der Durchführung jedes Teilexperiments kann jedoch nicht durch eine Zustandsänderung erreicht werden. Emulab verwaltet ebenfalls welche Festplattenabbilder auf jedem Knoten installiert sind, und ersetzt ein gleiches Festplattenabbild normalerweise erst nach Freigabe eines Knotens. Um ein Aufspielen bei jeder Wiederverwendung zu erzwingen, wird deshalb os_setup um eine Kommandozeilen-Option erweitert, die, unabhängig vom installierten Festplattenabbild, zu einem erneuten Aufspielen

führt.

# Kapitel 7

# Evaluation

Zur effizienten Unterstützung von Experimentläufen wurden im vorangegangenen Kapitel zwei Möglichkeiten vorgestellt, die gesamte Durchführungsdauer eines Experimentlaufs zu verkürzen. Um die Auswirkungen der Parallelisierung und Wiederverwendung von Knoten analysieren zu können, werden daher im Folgenden Messungen an zwei Szenarien durchgeführt und die Ergebnisse analysiert.

Die Durchführung findet einmal als Experimentlauf statt. Ein zweites Mal werden die entsprechenden Teilexperimente im Stapelbetrieb durchgeführt, und ein Vergleich der beiden Ablaufsteuerungen möglich.

## 7.1 Messumgebung und Metriken

**Szenarien**

In Abschnitt 2.1 wurden zwei Beispiele für die beiden hauptsächlichen Einsatzbereiche des Network Emulation Testbed der Universität Stuttgart gegeben und für beide Einsatzbereiche jeweils ein Szenario vorgestellt: Das Internet Szenario, das bereits als Beispiel in den letzten Kapiteln verwendet wurde, und das MANET Szenario. Beide Szenarien werden im Folgenden für die Evaluation verwendet.

Sie werden hierzu als zwei parametrisierte Experimentläufe angelegt. Für jeden Experimentlauf wird in der Szenariobeschreibung als Aktion für eine Wiederverwendung der Knoten die Rekonfiguration festgelegt.

Beide Szenarien werden zusätzlich im Stapelbetrieb ausgeführt. Für die Durchführung der Teilexperimente beider Szenarien werden im Stapelbetrieb entsprechende einzelne Batch-Experimente angelegt. Dazu werden die nicht-parametrisierten Szenariobeschreibungen (oTcl-Skripte) der Teilexperimente verwendet. Bei allen Teil- bzw. Batch-Experimenten wird während der Durchführungsphase lediglich ein Kommando ausgeführt, das die Durchführungsphase auf zwei Minuten begrenzt.

Für beide Szenarien werden im Folgenden Kurzbezeichnungen der entsprechenden Emulab-Experimentinstanzen verwendet. Der Name setzt sich aus drei Teilen zusammen, die mit einem Bindestrich verbunden werden: dem Namen des Szenarios (MaNET / iNET), der Durchführungsart (ER - Experimentlauf / BD - Stapelbetrieb) und der Seriennummer des Experiments.

Das Experiment „MaNET-ER-8“ ist somit das achte Teilexperiment des parametrisierten Experimentlaufs des MANET Szenarios.

### Messumgebung

Die Messumgebung ist das Network Emulation Testbed (NET). Von den 64 vorhandenen Cluster-Knoten werden 21 Knoten durch Emulab verwaltet. Von den 21 Knoten sind bei der Durchführung 17 Knoten als freie Ressourcen verfügbar. Während der Messungen werden keine weiteren Experimente durchgeführt.

Zur Durchführung der Batch-Experimente im Stapelbetrieb wird die Zeit für die Wiederholung nach einem fehlerhaften Swapin-Versuch vom Standardwert 15 Minuten auf 3 Minuten herabgesetzt. Damit wird die relativ kurze Durchführungsdauer der Experimente von nur 2 Minuten kompensiert, und eine Benachteiligung der gesamten Durchführungsdauer im Stapelbetrieb beim Vergleich zwischen Experimentläufen und Batch-Experimenten vermieden.

### Messwerte und Metriken

Während der Durchführung eines Experiments werden mehrere Messwerte erfasst. Die Messwerte werden dann herangezogen, um die gesamte Durchführung, den Verlauf der Teil- bzw. Batch-Experimente und die Belegung der physischen Ressourcen beschreiben und analysieren zu können.

Für alle Experimente wird eine Beschreibung des Ablaufs durch Betrachtung der folgenden Werte ermöglicht:

- Zeitpunkte der Swapin-, Durchführungs- und Swapout-Phasen,
- Zeitdauern der Swapin-, Durchführungs- und Swapout-Phasen,
- Zeitpunkte fehlerhafter Aktivierungen aufgrund fehlender Ressourcen.

Die Start- und Endzeitpunkte der Swapin-, Durchführungs- und Swapout-Phase werden direkt gemessen und daraus die Zeitdauern ermittelt. Fehlerhafte Aktivierungsversuche können ebenfalls direkt ermittelt werden. Die gesamte Durchführungsdauer wird durch den Beginn der Swapin-Phase des ersten Experiments und das Ende der Swapout-Phase des letzten Experiments festgelegt.

Um eine Beurteilung über den Grad der Parallelisierung geben zu können, wird eine Parallelität ermittelt, die, wie in Gleichung 7.1 angegeben, berechnet wird. Dazu wird über die Laufzeiten aller Experimente ($t_{\text{aktiv}}$) die Summe gebildet, und der Wert durch die gesamte Durchführungsdauer geteilt ($t_{\text{gesamt}}$). Die Parallelität gibt also den Mittelwert der gleichzeitig aktiven Experimente an. Ein Wert von 2 gibt an, dass während der gesamten Durchführungsdauer im Mittel zwei Experimente gleichzeitig aktiv sind.

$$\text{Parallelität } P = \frac{\sum_{\text{Exp.}} t_{\text{aktiv}}}{t_{\text{gesamt}}} \tag{7.1}$$

Ein Vergleich zwischen Experimentläufen ($t_{\mathrm{ER}}$) und Batch-Experimenten ($t_{\mathrm{BD}}$) wird durch den Speedup ermöglicht, der das Verhältnis der Durchführungsdauern zueinander angibt und nach Gleichung 7.2 berechnet wird.

$$\text{Speedup } S = \frac{t_{\mathrm{ER}}}{t_{\mathrm{BD}}} \tag{7.2}$$

Neben den Messwerten und abgeleiteten Werten für Experimente bzw. Experimentläufe werden zusätzlich Informationen über den Belegungszustand der physischen Knoten betrachtet. Ein Knoten ist belegt, wenn er nicht als freie Ressource zur Verfügung steht. Ein belegter Knoten ist aus zwei Gründen nicht als freie Ressource verfügbar. Entweder ist er für ein Experiment reserviert, oder er wird mit einem Festplattenabbild geladen. Für alle verwendeten physischen Knoten eines Experimentlaufs wird daher der Belegungszustand und bei einer Reservierung zusätzlich der Name des zugehörigen Experiments protokolliert.

Um eine Belastung der physischen Ressourcen ähnlich zur Parallelität für einen Experimentlauf beschreiben zu können, wird die „Knotenbelastung“ definiert. Sie beschreibt die mittlere prozentuale Belegung aller verfügbaren physischen Knoten bei Durchführung eines Experiments. Zur Berechnung wird die Summe der Belegungsdauern ($t_{\mathrm{belegt}}$) aller Knoten durch die Summe der gesamten Belegungszeit ($t_{\max}$) und die Anzahl aller verfügbaren physischen Knoten ($n$) geteilt. Die Belegungszeit $t_{\max}$ gibt dabei den Zeitraum von der Reservierung des ersten Knotens bis zur Freigabe des letzten Knotens nach einem Experiment und nach Aufspielen eines Festplattenabbildes an. Eine Belastung von 75% beschreibt dann, dass während einer Durchführung im Mittel dreiviertel aller verfügbaren Knoten belegt sind.

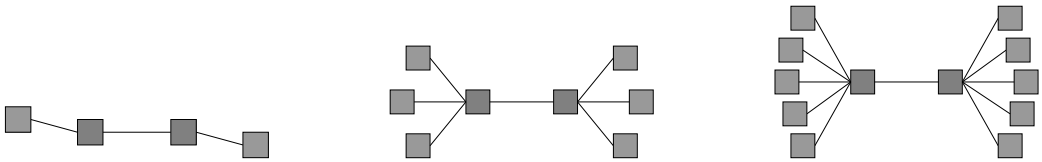
$$\text{Knotenbelastung } K = \frac{\sum_{\text{Knoten}} t_{\mathrm{belegt}}}{n \cdot t_{\max}} \tag{7.3}$$

## 7.2 Internet Szenario

Das Internet Szenario wurde in Abschnitt 2.1.3 als Beispiel eingeführt, um das Verhalten einer verteilten Anwendung unter verschiedenen Randbedingungen untersuchen zu können. Die Topologie wird durch ein „dumbbell“-Schema gebildet. Dazu werden mehrere Knoten an zwei Router angebunden. Zwischen beiden Routern besteht eine weitere Verbindung, die einen Flaschenhals bei Kommunikation der angebundenen Knoten darstellen kann.

Um verschiedene Randbedingungen zu erhalten, wird das Szenario über die Anzahl der angebundenen Knoten auf beiden Seiten und die Bandbreite des Flaschenhalses variiert. Die Anzahl der Knoten auf jeder Seite beträgt 1, 3 und 5 Knoten (Abb. 7.1). Die verwendeten Bandbreiten liegen bei 10 Mbps, 15 Mbps und 20 Mbps.

Aus den entstehenden neun Teilexperimenten ergibt sich dann die Knotenanzahl der Teilexperimente aus Tab. 7.1. Das zur Evaluation verwendete Szenarioskript ist in Abschnitt 6.2.3 angegeben.



(a) 4 Knoten (b) 8 Knoten (c) 12 Knoten

Abbildung 7.1: Topologien des parametrisierten Internet Szenarios

| Teilexperiment | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Anzahl Knoten | 4 | 4 | 4 | 8 | 8 | 8 | 12 | 12 | 12 |

Tabelle 7.1: Anzahl der Knoten der Teilexperimente des Internet Szenarios

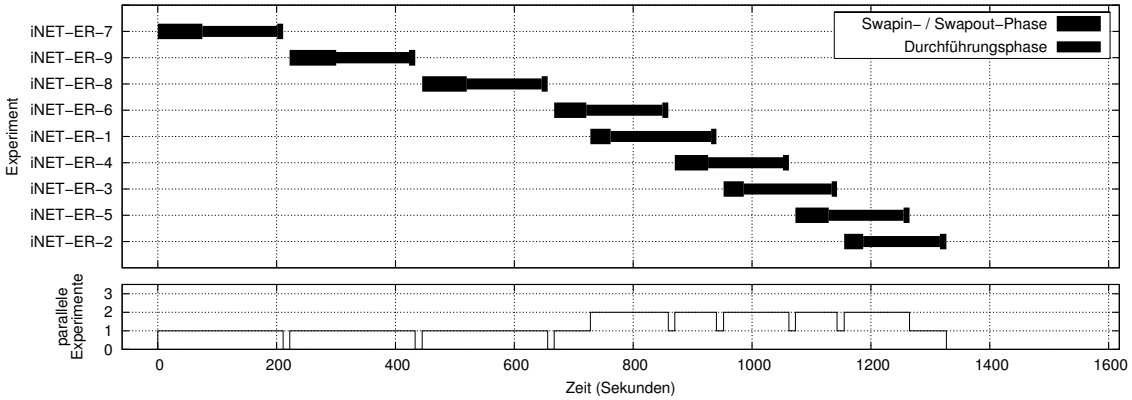
**Internet Szenario als Experimentlauf**

Der Ablauf der Teilexperimente des Internet Szenarios wurde als Beispiel der Parallelisierung in Abschnitt 6.2.3 beschrieben und ist in Abb. 7.2(a) nochmals dargestellt. Im oberen Teil der Abbildung ist der Ablauf der Teilexperimente während der Durchführung des Meta-Experiments dargestellt. Im unteren Teil ist die Anzahl der gleichzeitig aktiven Teilexperimente abgebildet. Zu Beginn der Durchführung werden zunächst alle drei Teilexperimente mit 12 Knoten nacheinander durchgeführt (iNET-ER-7, iNET-ER-9, iNET-ER-8). Anschließend wird die Menge der reservierten Knoten zur parallelen Durchführung jeweils eines Teilexperiments mit 8 Knoten und eines Teilexperiments mit 4 Knoten aufgeteilt. Die Durchführung findet dann für jeweils zwei Teilexperimente parallel statt (z.B. iNET-ER-6 und iNET-ER-1).

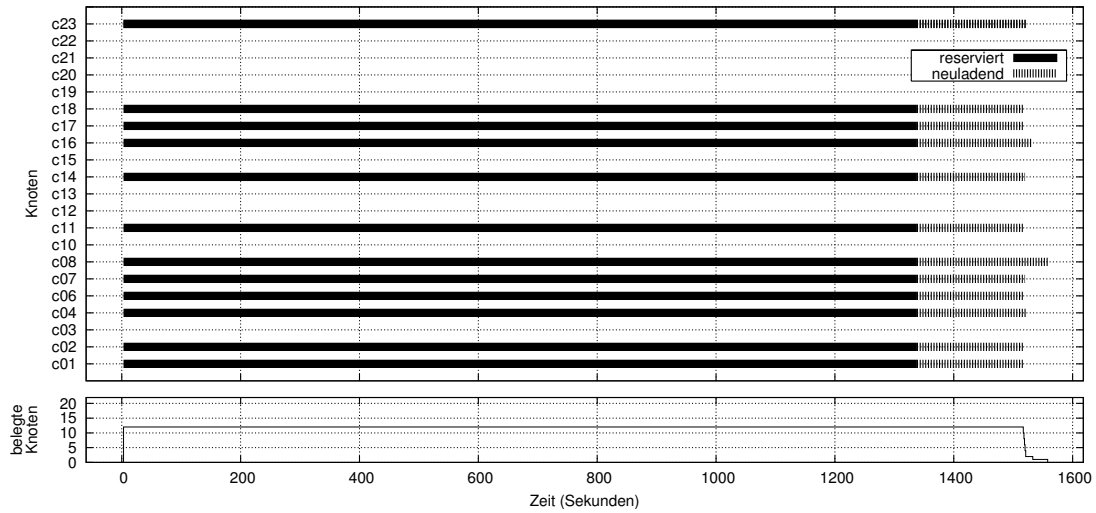
Die gesamte Durchführungsdauer des Meta-Experiments beträgt auf diese Weise 1327 Sekunden. Bei der Aktivierung der Teilexperimente treten keine Fehler auf, da die einmalige Reservierung der Knoten eine ausreichende Ressourcenverfügbarkeit garantiert. Die Teilexperimente laufen im Mittel 198 Sekunden. Auf die drei Phasen eines Experiments aufgeteilt, werden für die Swapin-Phase im Mittel 55s aufgewendet, die Durchführungsphase benötigt im Mittel 133s und die Swapout-Phase 10s. Die leicht erhöhte Dauer der Durchführungsphase von 133s gegenüber den in der Szenariobeschreibung festgelegten 120s ist durch die Serialisierung der Swapin- und Swapout-Vorgänge zu erklären. So findet z.B. der Swapout-Vorgang von iNET-ER-1 erst statt, nachdem der Swapin-Vorgang von iNET-ER-4 abgeschlossen wurde.

Für das Internet Szenario wird nach Gleichung 7.1 eine Parallelität von 1,34 berechnet. Während der gesamten Durchführung sind also im Mittel 1,34 Teilexperimente gleichzeitig aktiv. Durch das Verfahren der Parallelisierung könnte theoretisch ein Wert von 1,5 erreicht werden (3 Teilexperimente seriell, dann 3 mal jeweils 2 Teilexperimente parallel). Die aus Gründen der Lastverteilung vorgenommene Serialisierung der Swapin- und Swapout-Vorgänge verlängert die gesamte Durchführungsdauer jedoch, so dass der rechnerische Wert nicht erreicht werden kann.

In Abb. 7.2(b) ist für alle physischen Knoten deren Belegungszustand aufgeführt. Die Abbildung besteht ebenfalls aus zwei Teilen. Im unteren Teil ist die Anzahl der belegten Knoten




(a) Ablauf



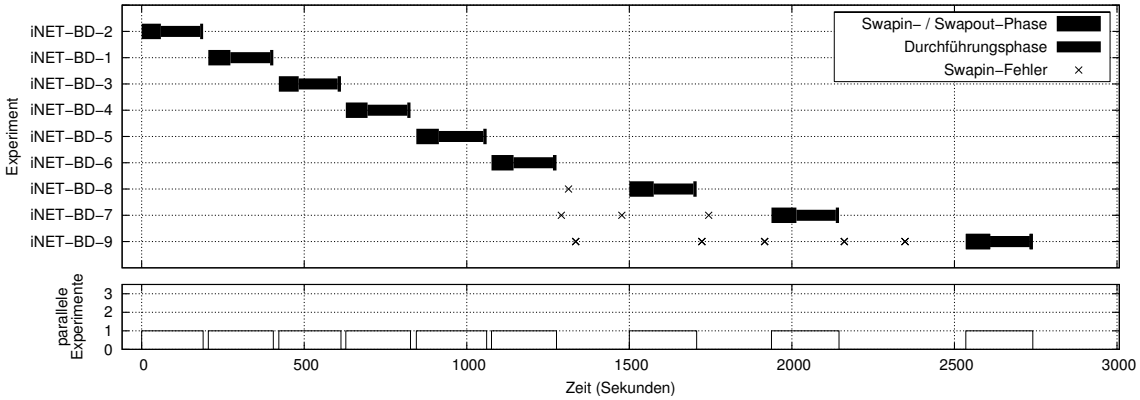

(b) Knotenbelegung

Abbildung 7.2: Ablauf des Internet Szenarios als Experimentlauf

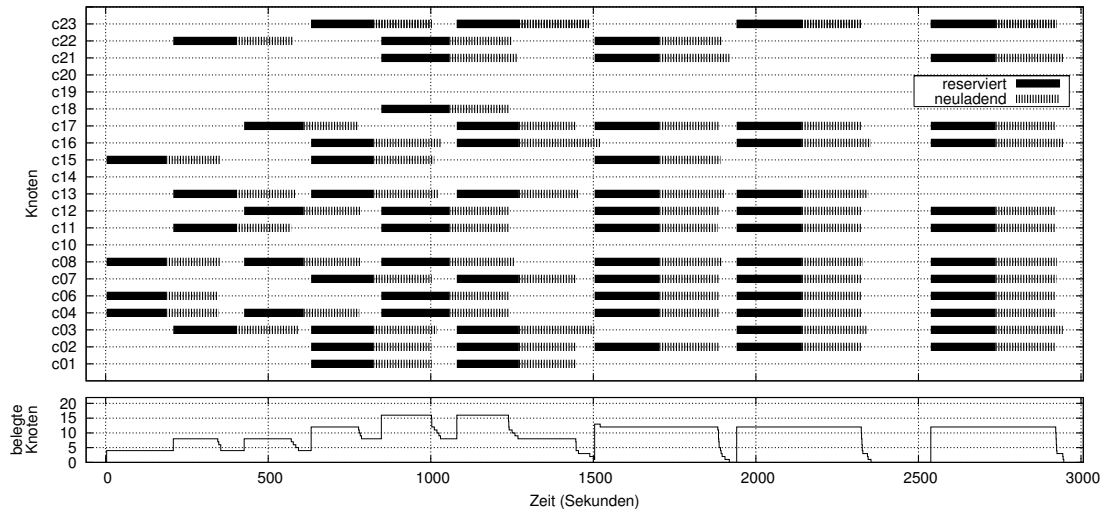
zu einem Zeitpunkt dargestellt. Im oberen Teil ist dargestellt, wann Knoten Experimenten zugeordnet sind und wann sie durch ein Aufspielen eines Festplattenabbildes ebenfalls nicht als freie Ressourcen zur Verfügung stehen. Der Vorgang des Aufspielens eines Festplattenabbildes wird im Folgenden auch als Neuladen bezeichnet.

Das Verfahren zur Parallelisierung von Experimentläufen führt zur einmaligen Belegung der 12 benötigen Knoten des größten Teilexperiments. Die Knotenbelastung beträgt 69%, d.h. die 17 verfügbaren Knoten sind während der gesamten Durchführungsdauer im Mittel zu 69% belegt. Davon sind sie während 61% der Laufzeit reserviert und während 8% durch Neuladen belegt.

Reservierte Knoten eines Experimentlaufs sind nicht durchgehend Teilexperimenten zugeordnet, sondern auch zeitweise dem Meta-Experiment. Während der Durchführung des Internet Szenarios sind die 12 reservierten Knoten zu 90% einem Teilexperiment zugeordnet. In der restlichen Zeit (10%) können die Knoten entweder nicht einem Teilexperiment zugeordnet werden, z.B. während der Durchführung des letzten Teilexperiments, oder die Knoten sind kurzzeitig




(a) Ablauf




(b) Knotenbelegung

Abbildung 7.3: Ablauf des Internet Szenarios im Stapelbetrieb

zwischen zwei Teilexperimenten dem Meta-Experiment zugeordnet.

### Internet Szenario im Stapelbetrieb

In Abb. 7.3(a) ist der Ablauf der Batch-Experimente bei Durchführung des Internet Szenarios im Stapelbetrieb dargestellt. Die Reihenfolge bei Durchführung mehrerer Batch-Experimente wird vom System nicht direkt festgelegt. Sie wird aber z.B. durch die Reihenfolge des Anlegens der Batch-Experimente und der damit verbundenen Reihenfolge von Ergebnissen bei Datenbankabfragen beeinflusst. Während der Durchführung der Batch-Experimente wird die Reihenfolge zusätzlich durch fehlerhafte Swapin-Versuche beeinflusst.

Zur Durchführung des Internet Szenarios werden alle Batch-Experimente zum gleichen Zeitpunkt gestartet. Da die Experimente in der Reihenfolge der Seriennummer angelegt wurden, werden zunächst Experimente mit wenigen Knoten ausgeführt (iNET-BD-1 bis iNET-BD-6), und danach die Experimente mit 12 Knoten (iNET-BD-7, iNET-BD-8, iNET-BD-9). Die ge-

samte Durchführungsdauer ist dann mit 2741s deutlich höher als bei Durchführung des Internet Szenarios als Experimentlauf. Durch fehlende Ressourcen treten im Ablauf, wie in Abb. 7.3(a) erkennbar, bei Aktivierung einzelner Batch-Experimente zudem 9 Swapin-Fehler auf, so dass der Versuch zu einem späteren Zeitpunkt wiederholt wird.

Jedes Batch-Experiment benötigt im Mittel 202s zur Durchführung. Auf die drei Phasen eines Experiments aufgeteilt, werden im Mittel 69s (Swapin-Phase), 123s (Durchführungsphase), bzw. 10s (Swapout-Phase) aufgewendet. Dabei ist allerdings für die Swapin-Phase die Zeit für das Neuladen (das Aufspielen eines Festplattenabbildes) nicht inbegriffen.

In Emulab wird *nach* jedem Experiment ein Standardfestplattenabbild aufgespielt. Ein Experiment wird daher, wenn es kein davon abweichendes Festplattenabbild verwendet, eine relativ kurze Swapin-Phase haben. Die physischen Knoten stehen allerdings nach der Durchführung eines Experiments erst nach dem Neuladen wieder als freie Ressource zur Verfügung. Der gesamte Aufwand bleibt daher, im Sinne der Belegungszeit der Knoten, bei Verwendung des Standardfestplattenabbildes gleich. Bestimmt ein Experimentator ein eigenes, spezielles Festplattenabbild, muss es während der Swapin-Phase geladen werden, so dass der Zeitaufwand steigt. Zur Durchführung des Internet Szenarios wird jedoch das Standardabbild verwendet, so dass der Neuladevorgang während der Swapin-Phase entfällt.

Daraus lassen sich auch der Ablauf und die Swapin-Fehler erklären. Batch-Experimente können direkt hintereinander ausgeführt werden, wenn nach Beendigung eines vorigen Experiments genügend freie Knoten verfügbar sind. Im Internet Szenario ist dies z.B. bei iNET-BD-2 und iNET-BD-1 der Fall. Nach Durchführung von iNET-BD-1 werden alle reservierten Knoten neu geladen, da aber gleichzeitig genügend weitere Knoten verfügbar sind, kann das Experiment iNET-BD-2 umgehend aktiviert werden. Andereseits sind nach Durchführung des Experiments iNET-BD-7 zunächst nicht genügend Knoten als freie Ressource verfügbar, und die Aktivierung des folgenden Experiments iNET-BD-9 schlägt zweimal fehl. Erst nach Beendigung des Neuladens der Knoten des vorigen Experiments ist daher ein Swapin-Versuch für das Experiment iNET-BD-9 erfolgreich.

Da Batch-Experimente eines Benutzers nur seriell ausgeführt werden, finden während der gesamten Durchführung des Szenarios keine Experimente parallel statt. Die Parallelität berechnet sich zu 0,66. Der intuitiv erwartete Wert von 1 wird nicht erreicht, da die Zeit zwischen einzelnen Batch-Experimenten ebenfalls einbezogen wird, und eine direkte Hintereinanderausführung der Batch-Experimente nicht immer möglich ist.

Das Belegungsschema der Knoten ist in Abb. 7.3(b) dargestellt. Im Stapelbetrieb ergibt sich aus der Arbeitsweise, die Knoten nach jedem Experiment neu zu laden, ein gänzlich anderes Bild gegenüber Experimentläufen. Die Ressourcenauswahl findet für alle Batch-Experimente unabhängig voneinander statt, und die Knoten werden aus den jeweils verfügbaren zufällig ausgewählt. Die Knotenbelastung beträgt beim Internet Szenario im Stapelbetrieb 55%, davon sind alle Knoten im Mittel zu 28% reserviert und zu 27% durch Neuladevorgänge belegt.

### Vergleich und Schlussfolgerungen

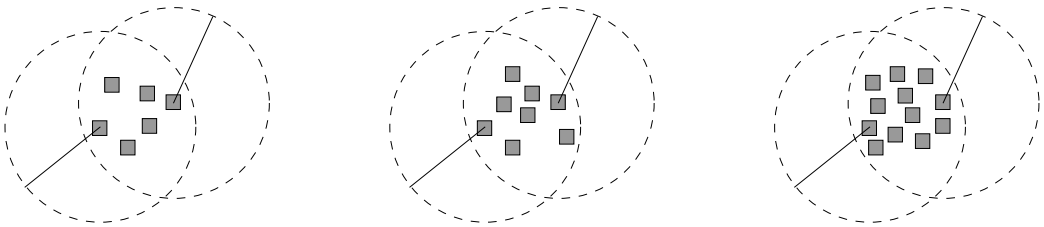
Die unterschiedliche Art der Durchführung von Experimentläufen und Batch-Experimenten führt zu deutlich verschiedenen Messwerten. Vergleicht man die Laufzeiten beider Durchführungsarten ergibt sich ein Speedup von 2,07 des Experimentlaufs gegenüber dem Stapelbetrieb.

Während im Stapelbetrieb die einzelnen Experimente serialisiert durchgeführt werden, kann durch die Parallelisierung die Durchführungszeit des Experimentlaufs deutlich verkürzt werden. Durch die Rekonfiguration der Knoten werden sie nicht nach jedem Teilexperiment neu geladen, so dass daraus ein weiterer Vorteil entsteht. Weiterhin wird beim Stapelbetrieb nach einem fehlerhaften Swapin-Versuch eines Batch-Experiments erst nach 3 Minuten ein erneuter Aktivierungsversuch unternommen. Sind während dieses Zeitraums keine weiteren Swapin-Versuche anderer Batch-Experimente erfolgreich, verlängert sich die gesamte Durchführungsdauer deutlich.

Der Aufwand der Durchführungs- und Swapout-Phase beider Durchführungsarten ist nahezu identisch. Ohne Betrachtung des Zeitaufwands für das Neuladen von Knoten beim Stapelbetrieb, unterscheiden sich die Dauern der Swapin-Phasen ebenfalls nicht markant (ER: 55s / BD: 69s).

Die Knotenbelastung ist beim Experimentlauf mit 69% höher als im Stapelbetrieb mit 55%. Summiert man die Belegungsdauern aller Knoten beider Durchführungsarten, werden beim Stapelbetrieb alle Knoten zusammen für 27683s belegt, beim Experimentlauf für 18245s. Demnach ist die Knotenbelastung beim Experimentlauf zwar höher, die gesamte Belegungszeit aber geringer. Die Nutzung der Knoten ist beim Experimentlauf also insgesamt effektiver.

## 7.3 MANET Szenario



(a) 6 Knoten (b) 8 Knoten (c) 12 Knoten

Abbildung 7.4: Topologien des parametrisierten MANET Szenarios

Das MANET Szenario wurde bereits in Abschnitt 2.1.3 eingeführt. Das Szenario beinhaltet mehrere Teilnehmer, die drahtlos miteinender kommunizieren und ihren Standort verändern können. Die Mobilität der Teilnehmer führt dazu, dass sich Verbindungseigenschaften ändern und u.U. eine (direkte) Kommunikation nicht mehr möglich ist. Es wird für das Szenario angenommen, dass sich anfänglich alle Teilnehmer in gegenseitiger Kommunikationsreichweite befinden. Während eines Experiments verändern die Teilnehmer dann (z.B. nach dem Random Waypoint Mobilitäts-Modell) ihren Standort.

Um verschiedene Randbedingungen zu erzeugen, wird das Szenario durch zwei Parameter variiert. Durch einen Parameter und Werte von 6, 8 und 12 wird die Anzahl der Teilnehmer verändert. Die „Topologien“ bei Beginn eines Experiments werden dann durch Abb. 7.4 beschrieben.

| Teilexperiment | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Anzahl Knoten | 6 | 6 | 6 | 8 | 8 | 8 | 12 | 12 | 12 |

Tabelle 7.2: Anzahl der Knoten der Teilexperimente des MANET Szenarios

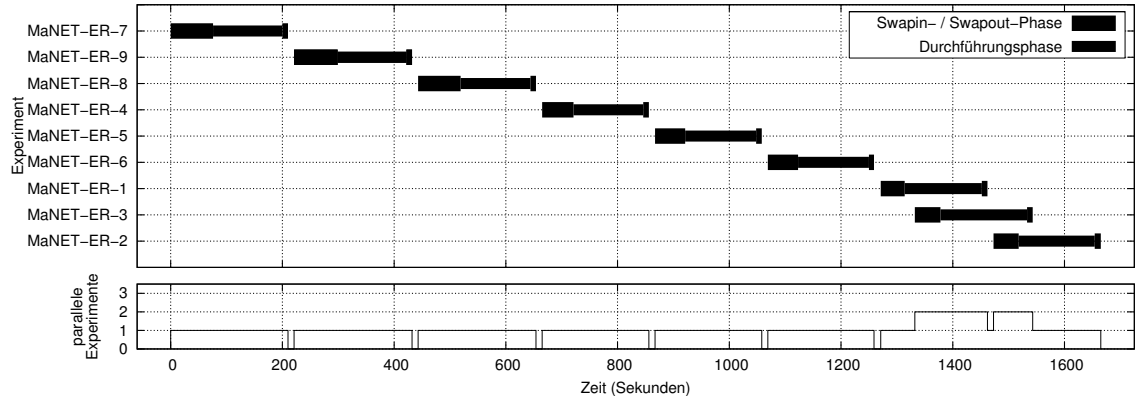



Abbildung 7.5: Ablauf des MANET Szenarios als Experimentlauf

Die Mobilität wird durch den zweiten Parameter variiert, der drei verschiedene Fortbewegungsgeschwindigkeiten der Teilnehmer nachbildet. Für die Evaluation wurden allerdings während der Durchführung der Experimente daraus resultierende Änderungen der Verbindungseigenschaften nicht vorgenommen. Der Parameter ist lediglich ein Beispiel einer möglichen Parametrisierung zur Evaluation verschiedener Randbedingungen.

Der Experimentlauf setzt sich damit ebenfalls aus neun Teilexperimenten zusammen. Die Anzahl der benötigten Knoten aller Teilexperimente ist in Tab. 7.2 aufgelistet. Das zur Evaluation verwendete Szenarioskript ist in Anhang A zu finden.

### MANET Szenario als Experimentlauf

Bei Durchführung der Teilexperimente des MANET Szenarios werden, wie in Abb. 7.5 dargestellt, zunächst die drei Teilexperimente mit der größten Anzahl benötigter Knoten nacheinander durchgeführt (MaNET-ER-7, MaNET-ER-9, MaNET-ER-8). Danach wird ein verbliebenes Teilexperiment mit maximaler Knotenanzahl ausgewählt (MaNET-ER-4). Nach der Reservierung der 8 Knoten für das Teilexperiment können keine weiteren Teilexperimente parallel ausgeführt werden (das kleinste verbliebene hat 6 Knoten). Daher werden auch alle restlichen Teilexperimente mit 8 Knoten zunächst nacheinander durchgeführt (MaNET-ER-5, MaNET-ER-6). Von den verbliebenen drei Teilexperimenten, die jeweils 6 Knoten benötigen, werden zwei parallel durchgeführt (MaNET-ER-1, MaNET-ER-3) und dann das letzte Teilexperiment MaNET-ER-2.

Während der Durchführung treten keine Swapin-Fehler auf, und die gesamte Durchführungsdauer beträgt 1665s. Im Mittel werden 59s für einen Swapin, 131s für die Durchführungsphase und 10s für einen Swapout benötigt. Die mittlere Laufzeit eines Teilexperiments beträgt demnach 200s. Die Parallelität des gesamten Experimentlaufs berechnet sich zu 1,08. Der Grad

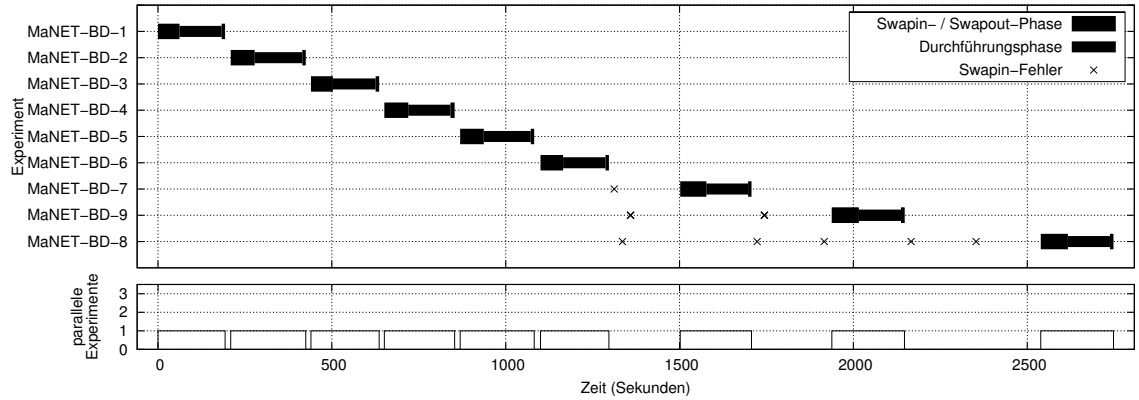



Abbildung 7.6: Ablauf des MANET Szenarios im Stapelbetrieb

der Parallelisierung ist demnach geringer als bei der Durchführung des Internet Szenarios als Experimentlauf.

Während der Durchführung liegt die Knotenbelastung bei 69%, der Anteil für den einmaligen Neuladevorgang am Ende des Experimentlaufs beträgt dabei 7%. Die reservierten 12 Knoten können nicht immer einem Teilexperiment zugewiesen werden. Bei der Durchführung der Teilexperimente mit 8 Knoten sind vier Knoten daher dem Meta-Experiment zugeordnet. Nach dem Ende von MaNET-ER-3 verbleiben während der Durchführung des letzten Teilexperiments (MaNET-ER-2) bis zum Ende der gesamten Durchführung ebenfalls 6 Knoten dem Meta-Experiment zugeordnet. Das Verhältnis zwischen aktiver Nutzung der Knoten in Teilexperimenten zu der Reservierung im Meta-Experiment liegt bei 78% zu 22%.

### MANET Szenario im Stapelbetrieb

Das MANET Szenario ist ebenfalls durch Anlegen einzelner Batch-Experimente erfolgt, und alle Batch-Experimente werden zum gleichen Zeitpunkt gestartet. Die Messergebnisse sind aufgrund der Ähnlichkeiten zum Internet Szenario und der Arbeitsweise der Ablaufsteuerung nahezu identisch zu den Ergebnissen des Internet Szenarios. Wie in Abb. 7.6 zu erkennen, werden alle Experimente nacheinander durchgeführt. Die Reihenfolge wird ebenfalls durch das Anlegen und Swapin-Fehler beeinflusst, und es werden alle Experimente mit steigender Knotenanzahl durchgeführt. Während der Durchführung treten 8 Swapin-Fehler auf.

Die gesamte Durchführungsdauer beträgt 2748s. Die mittlere Laufzeit eines Batch-Experiments liegt mit 204s um 2 Sekunden höher als beim Internet Szenario, was durch eine leicht längere Dauer der Durchführungsphase mit 125s im Mittel zu erklären ist. Ein Swapin dauert im Mittel 69s, ein Swapout im Mittel 10s.

Aus der Ähnlichkeit des Ablaufs ergeben sich auch ähnliche Messwerte für den Belegungsgrad der Knoten. Die Knotenbelastung liegt bei 60%, und alle Knoten sind im Mittel zu 31% reserviert und zu 29% durch Neuladevorgänge belegt.

### Vergleich und Schlussfolgerungen

Das MANET Szenario als Experimentlauf erreicht durch die Verteilung der Knotenzahlen der Teilexperimente nur eine Parallelität von 1,08. Dennoch ergibt sich trotz der daraus resultierenden längeren Laufzeit noch einen Speedup von 1,65 gegenüber der Durchführung im Stapelbetrieb.

Vergleicht man die Durchführungen der Experimentläufe des MANET Szenario und des Internet Szenarios, ist erkennbar, wie die vorgenommenen Änderungen für eine effiziente Unterstützung von parametrisierten Experimentläufen die Laufzeit und die Auslastung der Ressourcen beeinflussen. Während bei Durchführung des Internet Szenarios eine höhere Parallelität erreicht wird, kann durch die unterschiedliche Verteilung der Knotenanzahlen das MANET Szenario nur in einem geringeren Maße parallelisiert werden. Das MANET Szenario ist daher erst nach 1665s beendet, während für das Internet Szenario bei gleicher Anzahl der Teilexperimente nur 1327s benötigt werden.

Beim MANET Szenario sind zudem Knoten über längere Zeit dem Meta-Experiment zugeordnet, ohne aktiv für Teilexperimente eingesetzt werden zu können. Der Anteil der aktiven Nutzung beträgt beim MANET Szenario daher lediglich 78%, während beim Internet Szenario die Knoten während 90% der Laufzeit einem Teilexperiment zugeordnet sind.

## 7.4 Gleichzeitiges Durchführen beider Szenarien

Beide Szenarien, das Internet Szenario und das MANET Szenario, werden im Folgenden gleichzeitig durchgeführt. Die Durchführung erfolgt für einen Vergleich ebenfalls einmal als Experimentlauf und einmal im Stapelbetrieb. In beiden Fällen werden die beiden Meta-Experimente bzw. alle Batch-Experimente zum gleichen Zeitpunkt gestartet. Die Ablaufsteuerungen des Systems müssen daher die Durchführung beider Szenarien selbst koordinieren.

### Beide Szenarien als Experimentlauf

Die Experimentläufe der Szenarien können mit den verfügbaren Knoten nicht parallel ausgeführt werden. Beide Meta-Experimente benötigen 12 Knoten, es stehen aber nur 17 zur Verfügung. Daher werden die Meta-Experimente nacheinander ausgeführt, die Teilexperimente der Meta-Experimente allerdings wieder teilweise parallel.

Der Ablauf in Abb. 7.7 beider Szenarien entspricht durch die Serialisierung der Meta-Experimente daher den Abläufen aus den vorigen Abschnitten (vgl. Abb. 7.2(a), Abb. 7.6). Zwischen den Meta-Experimenten ist allerdings ein gewisser zeitlicher Abstand zu erkennen, der sich daraus ergibt, dass nach der Durchführung des ersten Szenarios auf alle verwendeten Knoten zunächst das Standardfestplattenabbild aufgespielt wird. Die gesamte Durchführungsdauer beträgt dann 3315s.

Die weiteren Messwerte sind durch die serielle Durchführung beider Experimentläufe nahezu Mittelwerte der bereits in den vorigen Abschnitten gegebenen Messwerte. Eine Swapin-Phase dauert im Mittel 60s, die Durchführungsphase 134s und die Swapout-Phase 10s. Ein Teilexperiment ist im Mittel nach 204s beendet. Bei Durchführung des Teilexperiments iNET-

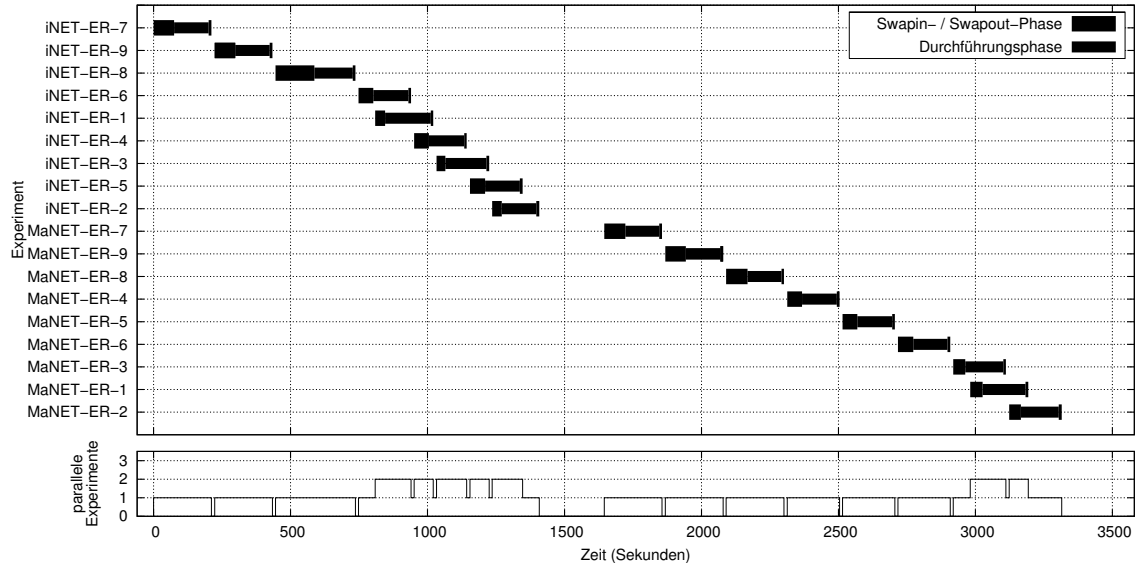



Abbildung 7.7: Ablauf beider Szenarien als Experimentlauf

ER-8 wurde allerdings eine überlange Swapin-Dauer von 142s gemessen, die durch einen fehlerhaften SSH-Aufruf entstand, der den weiteren Ablauf allerdings nicht beeinflusste.

Die Parallelität ist 1,11 und stellt den über die Durchführungsdauern gewichteten Mittelwert der beiden Meta-Experimente zuzüglich der Zeit für das Neuladen der Knoten zwischen den Meta-Experimenten dar. Die Knotenbelastung liegt bei 70%. Dabei sind die Knoten zu 62% reserviert und zu 8% durch ein Neuladen belegt. Der Anteil der aktiven Nutzung reservierter Knoten liegt bei 84%, und während 16% der Zeit sind die Knoten dem Meta-Experiment zugeordnet.

### Beide Szenarien im Stapelbetrieb

Um eine parallele Durchführung der beiden Szenarien im Stapelbetrieb zu ermöglichen, werden verschiedene Benutzerkennungen für die Batch-Experimente verwendet. Alle Batch-Experimente des Internet Szenarios werden von einem Benutzer angelegt, die Batch-Experimente des MANET Szenarios von einem anderen. Damit können jeweils ein Experiment des Internet Szenarios und ein Experiment des MANET Szenarios parallel durchgeführt werden. Experimente des gleichen Szenarios werden jedoch nacheinander durchgeführt.

Der Ablauf nach gleichzeitigem Starten aller Batch-Experimente ist in Abb. 7.8 dargestellt. Alle Batch-Experimente wurden in der Reihenfolge der Seriennummer angelegt. Eine andere Reihenfolge beim Anlegen würde die Messergebnisse jedoch qualitativ nicht beeinflussen. Zunächst laufen Batch-Experimente mit einer geringen Knotenanzahl beider Szenarien parallel (z.B. iNET-BD-2, MaNET-BD-1). Nach Durchführung der kleinen Batch-Experimente, die meist direkt hintereinander gestartet werden können, ist die Knotenanzahl für die übrigen Batch-Experimente größer und es treten vermehrt Swapin-Fehler auf. Die Knoten der vorigen Experimente sind zwar nicht reserviert, beim Versuch ein weiteres zu aktivieren, werden sie

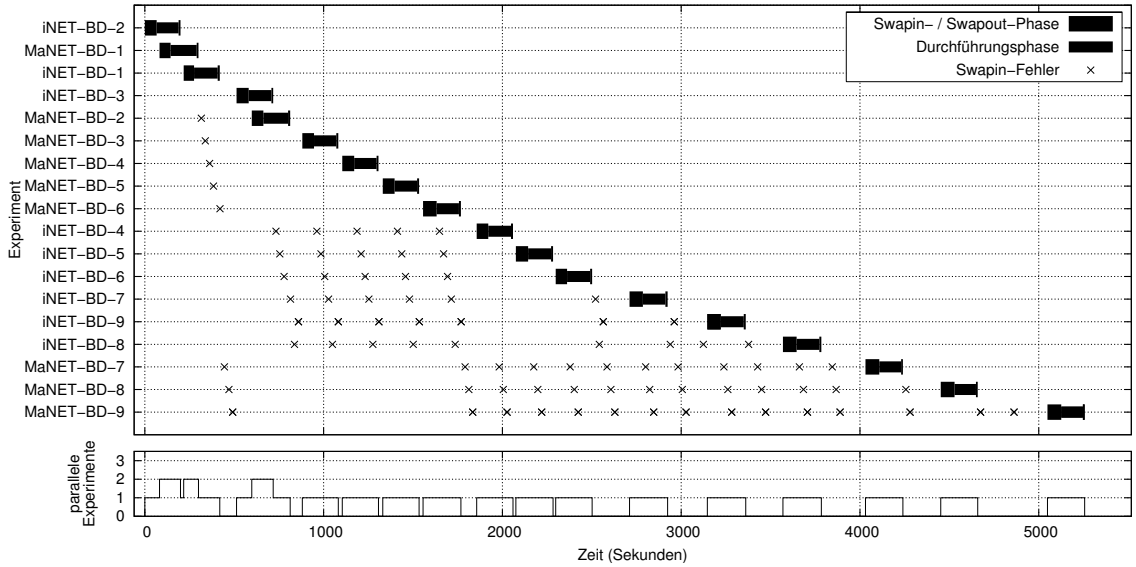



Abbildung 7.8: Ablauf beider Szenarien im Stapelbetrieb

aber noch neu geladen und sind nicht als freie Ressource verfügbar. Daher verzögert sich die gesamte Durchführungsdauer erheblich und beträgt dann 5257s. In insgesamt 82 Fällen schlagen dabei Swapin-Versuche fehl. Die Swapin-Phase dauert im Mittel 69s, die Swapout-Phase 11s und die Durchführungsphase 128s. Alle Batch-Experimente sind daher nach einer Laufzeit von jeweils 208s beendet.

Die Parallelität ist durch gleichzeitiges Durchführen der kleinen Batch-Experimente mit einem Wert von 0,71 gegenüber den unabhängigen Durchführungen der Szenarien im Stapelbetrieb leicht gestiegen. Die Knotenbelastung beträgt 63%, mit einem Anteil von 31% für Neuladevorgänge und einem Anteil von 32% für Reservierung in Batch-Experimenten.

#### Vergleich und Schlussfolgerungen

Obwohl die beiden Experimentläufe aufgrund zu geringer Ressourcenverfügbarkeit nur nacheinander ablaufen können, führt der Vergleich der Durchführungsdauern der Szenarien als Experimentläufe gegenüber dem Stapelbetrieb noch zu einem Speedup von 1,59 zugunsten der Experimentläufe. Bei genügend Ressourcen (24 Knoten) würden auch die Meta-Experimente und nicht nur die jeweiligen Teilexperimente parallelisiert, und der Speedup einen noch höheren Wert aufweisen. Die Durchführung im Stapelbetrieb wird vor allem durch das fortwährende Neuladen der Knoten und die dadurch entstehenden Aktivierungsfehler verlängert.

## 7.5 Vergleich der Wiederverwendungsarten

In Abschnitt 6.2.4 wurde ein Befehl vorgestellt, der die Aktion bei einer Wiederverwendung von Knoten zwischen Teilexperimenten steuert (tb-exprun-onreclaim). Mögliche Optionen sind

| | Swapin-Phase | | | Swapout-Phase | | |
|---|---|---|---|---|---|---|
| | Minimum | Maximum | Mittelwert | Minimum | Maximum | Mittelwert |
| reconfigure | 24s | 99s | 58,07s | 9s | 11s | 9,94s |
| reboot | 110s | 128s | 119,81s | 9s | 11s | 9,94s |
| reload | 209s | 262s | 237,75s | 9s | 198s | 21,75s |

Tabelle 7.3: Dauer der Swapin-Phase bei reconfigure, reboot, reload

dabei Rekonfiguration (reconfigure), Neustart (reboot) und Neuladen durch Aufspielen eines Festplattenabbildes (reload). Dabei wurde festgehalten, dass die jeweilige Aktion zu unterschiedlichem Aufwand führt. Da die Experimentläufe bisher eine Rekonfiguration durchführten, wurde der unterschiedliche Aufwand noch nicht ermittelt. Um den Aufwand zu ermitteln, werden deshalb drei weitere Experimentläufe durchgeführt.

Jeder der drei Experimentläufe wird durch 17 Teilexperimente gebildet. Die Teilexperimente werden durch eine steigende Anzahl von Knoten variiert, wobei mit einem Teilexperiment aus einem Knoten begonnen wird und das größte Teilexperiment 17 Knoten enthält. Jeder der drei Experimentläufe verwendet dann jeweils eine der Optionen zur Wiederverwendung von Knoten. Ziel ist es, den Aufwand der Optionen reconfigure, reboot und reload vergleichen zu können. Dazu wird die Dauer der Swapin-Phase gemessen.

Der Ablauf der Experimentläufe wird allerdings nicht eingehend betrachtet. Für die Experimentläufe ergibt sich jedoch, dass nach Durchführung des größten Teilexperiments jeweils das nächstgrößte und das kleinste parallel durchgeführt werden. Die Parallelität eines gesamten Experimentlaufs sollte demnach ungefähr 1,89 sein, da das erste Teilexperiment nicht parallelisiert wird (gemessen: 1,75). Der Ablauf und die Parallelität haben allerdings keinen Einfluss auf die Messergebnisse, da Swapin- und Swapout-Vorgänge serialisiert werden. Die Knotenbelastung ist, da alle zur Verfügung stehenden Knoten verwendet werden, bei allen drei Experimentläufen 99%.

### Vergleich des Aufwands der Wiederverwendungsarten

Abb. 7.9 zeigt die gemessenen Werte der Swapin-Phasen, Tab. 7.3 enthält ausgewählte Werte einzelner Swapin- und Swapout-Phasen. Zur Auswertung werden nur Teilexperimente zwischen 1 und 16 Knoten verwendet, da das erste, größte Teilexperiment mit 17 Knoten nicht den kompletten Zyklus der Initialisierung durchläuft.

Vergleicht man alle drei Arten der Wiederverwendung, ist der unterschiedliche Aufwand in Tab. 7.3 klar zu erkennen. Eine Rekonfiguration dauert im Mittel 58s, ein Neustart 120s, und das erneute Aufspielen eines Festplattenabbildes vor jeder Durchführung eines Teilexperiments 238s. Betrachtet man die Messkurven der drei Wiederverwendungsarten in Abb. 7.9 fällt auf, dass alle in Abhängigkeit von der Knotenanzahl einen ansteigenden Werteverlauf aufweisen. Bei Neustart und Neuladen ist der zusätzliche Aufwand nur gering, bei Rekonfiguration ist der Zusammenhang zwischen Knotenanzahl und zusätzlichem Aufwand ausgeprägter.

Eine Rekonfiguration der Knoten zwischen zwei Teilexperimenten führt zum geringsten

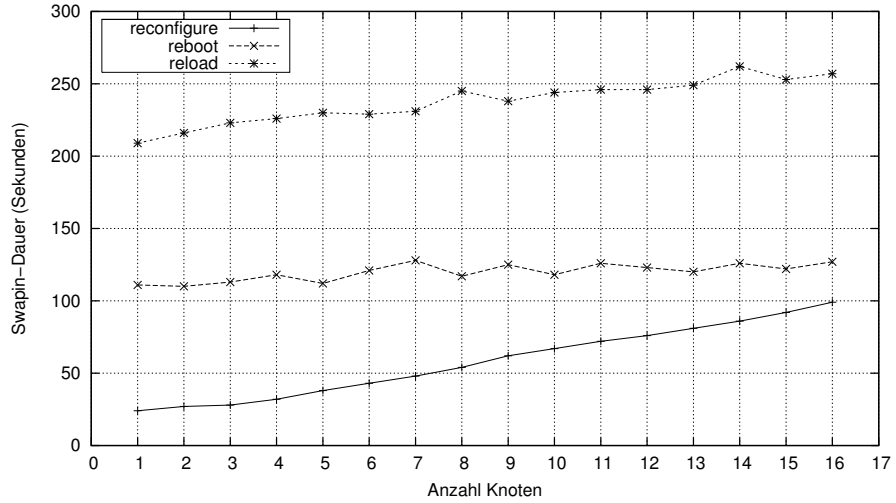



Abbildung 7.9: Dauer für Swapin bei steigender Knotenzahl

Aufwand. Der Mittelwert liegt bei 58s. Die Messwerte weisen mit einem Minimum von 24s (1 Knoten) und einem Maximum von 99s (16 Knoten) jedoch eine relativ breite Streuung auf. Während der Rekonfiguration finden zentrale Aufgaben auf dem Steuerungsrechner und eine dezentrale Rekonfiguration der Knoten parallel statt. Insbesondere findet die zentrale Konfiguration des Verbindungsnetzes parallel zur Rekonfiguration der Knoten statt. Eine Auswertung der Log-Dateien der Teilexperimente ergab, dass bei steigender Knotenzahl die Konfiguration des Verbindungsnetzes überdurchschnittlich lange dauert, und daher die gesamte Swapin-Phase verlängert wird. Die Ursache ist allerdings unbekannt. Es kann lediglich vermutet werden, dass die parallele Durchführung der beiden Aufgaben ein Fehlverhalten verursacht, das zu den Verzögerungen führt.

Daraus ergibt sich, dass eine Rekonfiguration nicht in allen Fällen die beste Möglichkeit ist, die Zeit zwischen zwei Teilexperimenten möglichst effizient zu gestalten. Der Schnittpunkt der reconfigure-Messkurve mit der reload-Messkurve liegt bei einem geschätzten Wert von 23 Knoten. Der Schnittpunkt der reconfigure-Messkurve mit der reload-Messkurve liegt bei ca. 95 Knoten. Da allerdings das Verhalten aufgrund eines Fehlverhaltens der Software entsteht, kann darauf gehofft werden, den Verlauf der reconfigure-Messkurve den beiden anderen angleichen zu können.

Bei Wiederverwendung der Knoten mittels Neustart steigen die Messwerte ebenfalls leicht mit zunehmender Anzahl der Knoten an. Der Anstieg ist aber geringer als die bei Rekonfiguration. Ein Swapin dauert im Mittel 120s mit einem Minimum von 110s (2 Knoten) und einem Maximum von 128s (7 Knoten). Die Abhängigkeit der Swapin-Dauer ergibt sich daraus, dass die Kommandos zum Neustart an die Knoten sowie zunehmende Anfragen an TFTP- und DHCP-Server den Steuerungsrechner mehr belasten.

Die Messkurve bei Neuladen steigt ebenfalls leicht mit zunehmender Knotenanzahl an. Ein Swapin dauert im Mittel 238s mit einem Minimum von 209s (1 Knoten) und einem Maximum von 262s (14 Knoten). Die zunehmende Dauer einer Swapin-Phase bei Neuladen kann ebenfalls erklärt werden. Eine Swapin-Phase ist abgeschlossen, wenn alle Knoten ein neu-

es Festplattenabbild erhalten und konfiguriert haben. Zur Verteilung der Festplattenabbilder wird das Programm Frisbee verwendet [HSL+03]. Laden mehrere Knoten ein Festplattenabbild gleichzeitig, ist die Last im Netz durch Übertragung mittels Multicast unabhängig von der Zahl der Knoten. Jedoch führt dies dazu, dass Teile eines Festplattenabbilds doppelt übertragen werden müssen, da die Knoten normalerweise nicht zum exakt gleichen Zeitpunkt dem Multicast-Stream beitreten, und somit Anfragen zu Teilen eines Festplattenabbildes mehrfach auftreten.

#### Vergleich zu Batch-Experimenten

Vergleicht man die Swapin-Dauern der drei Experimentläufe mit den bei Batch-Experimenten gemessenen Werten, scheint eine Rekonfiguration zunächst nicht den gewünschten Effekt zu haben, die Zeit zwischen zwei Teilexperimenten zu verkürzen. Ein Swapin-Vorgang dauert für ein Batch-Experiment bei den durchgeführten Szenarien im Mittel 69s, während bei Experimentläufen 59s gemessen werden. Der Zeitgewinn scheint also eher gering zu sein.

Bei den Werten der Swapin-Dauer bei Batch-Experimenten ist allerdings die Zeit für das Neuladen eines Knotens nach einem Experiment nicht enthalten. Die Zeit für einen Neuladevorgang kann aus den Messwerten mit 185s im Mittel angegeben werden. Damit verlängert sich die reale Dauer zwischen der Freigabe eines Knotens nach einem Experiment in der Swapout-Phase und dem Ende der Swapin-Phase des folgenden Experiments auf 254s. Dieser Wert liegt damit über dem gemessenen Mittelwert von Experimentläufen bei Neuladen der Knoten (238s).

## 7.6 Zusammenfassung und Fazit

Die Durchführung der beiden Szenarien als Experimentlauf zeigen, dass die vorgenommenen Änderungen für eine effiziente Unterstützung von parametrisierten Experimentläufen erfolgreich sind. Bei den hier vorgestellten Experimentläufen sind die Laufzeiten im Vergleich zu den im Stapelbetrieb ermittelten Werten kürzer. Der Speedup liegt zwischen 1,59 und 2,07.

Dabei greifen beide realisierten Mechanismen zur Verkürzung der Durchführungsdauer. Besteht die Möglichkeit zur Parallelisierung, ist die gemessene Laufzeit geringer, selbst wenn ein Experimentlauf nur in geringerem Umfang parallelisiert werden kann, wie dies beim MANET Szenario der Fall ist.

Die Rekonfiguration der Knoten und die damit verbundene Vermeidung der Neuladevorgänge hilft zudem, die Zeitdauer zwischen zwei Teilexperimenten zu reduzieren. Der Zeitgewinn tritt vor allem bei der parallelen Durchführung der Szenarien zu Tage, bei der im Stapelbetrieb das Neuladen der Knoten und damit häufig auftretende Swapin-Fehler einen effizienten Ablauf nicht erlauben. Die gleichzeitige Durchführung zeigt ebenfalls, dass sogar bei einer seriellen Ausführung der beiden Experimentläufe gegenüber der Ausführung der Batch-Experimente im Stapelbetrieb eine kürzere Laufzeit erreicht wird.

Die Knotenbelastung ist bei allen Experimentläufen höher als im Stapelbetrieb. Die Knoten werden aber über einen geringeren Zeitraum belegt. Summiert man die Belegungszeiten aller beteiligten Knoten bei Durchführung eines Szenarios auf [1], werden bei Experimentläufen auf-

[1] entspricht dem Zähler des Bruchs zur Berechnung der Knotenbelastung in Gleichung 7.3

grund der kürzeren Laufzeit geringere Werte als im Stapelbetrieb berechnet. Die Nutzung der Knoten ist daher bei Experimentläufen insgesamt effizienter, obwohl sie durch das gewählte Verfahren zur Parallelisierung zeitweise nicht aktiv eingesetzt werden, sondern lediglich dem Meta-Experiment zugeordnet sind.

Um vergleichbare Messwerte zu erhalten, sind alle Experimente ohne weitere Belastung des Systems durch andere Experimente durchgeführt worden. Ein Aspekt konnte dabei nicht betrachtet werden. Alle Batch-Experimente bzw. Experimentläufe konnten umgehend nach dem Kommando zur Ausführung aktiviert werden. Dieser Fall ist im Allgemeinen nicht immer gegeben. Wenn z.B. nur eine geringe Ressourcenverfügbarkeit durch andere gleichzeitig aktive Experimente besteht, kann sich der Ausführungszeitpunkt verschieben.

In einem Experimentlauf wird zunächst das größte Teilexperiment mit der größten Knotenanzahl ausgeführt. Bei geringer Ressourcenverfügbarkeit ist daher eine Aktivierung erst möglich, wenn für dieses Teilexperiment genügend Ressourcen reserviert werden können. Im Stapelbetrieb werden Batch-Experimente einzeln und unabhängig voneinander durchgeführt. Es ist daher zu erwarten, dass Batch-Experimente mit geringer Knotenanzahl trotz geringer Ressourcenverfügbarkeit früher aktiviert werden können. Die Durchführung des größten Batch-Experiments eines Szenarios ist dann aber ebenfalls erst möglich, wenn nach Beendigung anderer Experimente genügend Knoten zur Verfügung stehen.

Damit ist zu erwarten, dass Teilergebnisse bei einer Durchführung mehrerer Batch-Experimente früher verfügbar sind als bei Experimentläufen, wenn ein Teil der Experimente eine geringe Knotenanzahl aufweist. Im Gegensatz dazu können nach der erfolgreichen Aktivierung eines Experimentlaufs alle Teilexperimente ohne weitere Verzögerungen ausgeführt werden, da genügend Knoten bereits während der Durchführung des ersten Teilexperiments reserviert werden. Die Analyse der Messergebnisse einer Menge von Experimenten, wie z.B. bei einer Evaluation verschiedener Randbedingungen, wird meist erst dann sinnvoll möglich sein, wenn alle Experimente durchgeführt wurden und die Ergebnisse in Relation zueinander betrachtet werden können. Die mögliche Verzögerung des Startzeitpunkts von Experimentläufen bei geringer Ressourcenverfügbarkeit stellt damit keinen Nachteil dar.

# Kapitel 8

# Resümee

In diesem Kapitel wird zunächst ein Ausblick und Hinweise für mögliche Erweiterungen gegeben, und abschließend die Ergebnisse der Arbeit rekapituliert.

## 8.1 Ausblick

Die Umsetzung der allgemeinen Anforderungen für eine Unterstützung von parametrisierbaren Experimentläufen in den Abschnitten 5.1 und 6.1 sind teilweise nur durch tief greifende Änderungen des Emulab-Systems zu erreichen, und wurden daher im Zuge der Diplomarbeit nicht alle umgesetzt. So wäre z.B. eine Planung von Experimentläufen in einem bestimmten Zeitraum (z.B. „über Nacht“) wünschenswert. Die Durchführung eines Experimentlaufs zu einem festgelegten Zeitpunkt ist jedoch nur mit einer globalen Ressourcenverwaltung zu erreichen. Reservierungen von Ressourcen müssen dazu über einen bestimmten Zeitraum vom System geplant und garantiert werden. In Emulab sind dazu grundlegende Änderungen nötig, da eine globale Ressourcenverwaltung für Experimente nicht existiert und Änderungen alle Durchführungsarten beträfen.

In den eingeführten Veränderungen zur Unterstützung parametrisierter Experimentläufe sind aber Erweiterungen und zukünftige Entwicklungen in zwei Bereichen zu erkennen. Eine Unterstützung von Abhängigkeiten für Experimente und die damit verbundene Ergebnisverwaltung stellen Benutzern Möglichkeiten bereit, Evaluationen durchzuführen und dabei Ergebnisse von Teilexperimenten zur automatischen Definition und Durchführung weiterer Teilexperimente zu nutzen. Ferner führt das vorgestellte Verfahren zur Parallelisierung, den Ergebnissen der Evaluation nach zu urteilen, zum gewünschten Erfolg, kann aber weiter verfeinert werden. Für beide Erweiterungen werden im Folgenden Hinweise für eine Realisierung gegeben.

#### Unterstützung von Abhängigkeiten zwischen Teilexperimenten

Eine Abhängigkeit zwischen Teilexperimenten ist gegeben, wenn die erfolgreiche Durchführung eines oder mehrerer Teilexperimente vorausgesetzt wird, um deren Ergebnisse zur Definition und Durchführung eines weiteren verwenden zu können. Um Abhängigkeiten zwischen Experimenten unterstützen zu können, sind jedoch viele Aspekte zu betrachten, und eine Unterstützung führt daher zu einem hohen Realisierungsaufwand. Ob Benutzer einen regelmäßigen

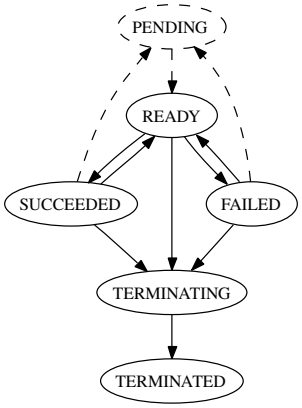



Abbildung 8.1: Erweiterter Zustandsgraph für abhängige Teilexperimente

Einsatz von Experimentläufen mit Abhängigkeiten anstreben, der diesen Aufwand rechtfertigt, ist aber derzeit nicht eindeutig zu beantworten.

Für eine Unterstützung von Abhängigkeiten müssen zunächst Möglichkeiten bestehen, Abhängigkeiten zu definieren. Dazu könnten in einem Szenarioskript z.B. Variablen in einem eigenen Namensraum angelegt werden. Eine zweite Möglichkeit wäre es, den Parser um eine Klasse zu erweitern, durch die Zugriffe auf Ergebnisse durch Objekte möglich sind. Ist der Zugriff dann während des Parsens nicht möglich, da die entsprechenden Ergebnisse fehlen, können damit fehlende Ergebnisse von abhängigen Experimenten erkannt werden. Die Ergebnisse von Teilexperimenten müssen während des Parsens also bekannt sein. Bereits vorhandene Ergebnisse könnten dazu vor jedem Aufruf des Parsers in das oTcl-Skript eingefügt werden. Wird beim Parsen festgestellt, dass noch Experimente durchzuführen sind, jedoch alle Abhängigkeiten nicht auflösbar sind, bestehen zyklische Abhängigkeiten zwischen Experimenten.

Ergebnisse von Experimenten – Messergebnisse oder davon abgeleitete Werte – müssen für einen Zugriff während des Parsens nach der Durchführung eines Experiments dem System übergeben werden können. Dazu muss eine Ergebnisverwaltung realisiert werden. Die Übermittlung von Ergebnissen an das System könnte durch Erweiterungen der Programme realisiert werden, die derzeit zur Übermittlung der Rückgabewerte der Startkommandos dienen [Gro05h].

Abhängigkeiten eines Experimentlaufs beeinflussen auch die Reihenfolge der Durchführung. Daher muss ein Ablaufgraph bestimmt werden. Sind nicht alle Abhängigkeiten aufgelöst, d.h. werden zur Durchführung noch Ergebnisse anderer Experimente benötigt, muss der Ablaufgraph dynamisch nach jedem Experiment erstellt werden. Die Experimentlauf-Steuerung kann nur Teilexperimente durchführen, deren Abhängigkeiten aufgelöst werden konnten. Für die Experimentlauf-Steuerung muss also erkennbar sein, ob Teilexperimente bereits durchgeführt werden können. Dies kann z.B. für alle Teilexperimente durch einen weiteren Zustand (PENDING) in der TE-Zustandsmaschine erreicht werden (siehe Abb. 8.1). Alle Teilexperimente, deren Abhängigkeiten noch nicht aufgelöst werden konnten, sind dabei im Zustand PENDING. Werden Abhängigkeiten nach der Durchführung eines Teilexperiments und erneuten parsen aufgelöst, so entspricht dies einem Zustandübergang von PENDING nach READY.

**Dynamische Anpassungen des Parallelisierungsgrads**

Das gewählte Verfahren der Parallelisierung stellt eine Möglichkeit dar, das hauptsächliche Problem bei gleichzeitiger Durchführung mehrerer Teilexperimente zu lösen: Die parallele Durchführung von möglichst vielen Teilexperimenten zu ermöglichen und gleichzeitig eine sinnvolle Beschränkung der Ressourcenauslastung zu realisieren. Dabei wird im vorgestellten Ansatz die Ressourcenauslastung durch die Anzahl der Knoten des größten Teilexperiments beschränkt. Das Verfahren garantiert ebenfalls, dass nach einer erfolgreichen Aktivierung des ersten Teilexperiments alle weiteren ohne Unterbrechung aufgrund fehlender Ressourcen durchgeführt werden können. Die Durchführung findet dabei (wenn möglich) parallel statt.

Die reservierten Knoten während der Durchführung können dabei als eine Partition der verfügbaren physischen Knoten betrachtet werden. Die Größe der Partition ist beim gewählten Verfahren statisch und vorab festgelegt.

Eine Erweiterung wäre es, die Größe der Partition, bzw. die Anzahl der Knoten, in Abhängigkeit der Ressourcenauslastung zu verändern. Bei Aktivierung wird einem Experimentlauf eine Anzahl von Knoten zur Verwendung bereitgestellt, z.B. die Knotenanzahl des größten Teilexperiments. Während der Durchführung und bei einer steigenden Ressourcenauslastung durch andere Experimente werden dem Experimentlauf dann Knoten nach dem Ende eines Teilexperiments entzogen. Die Partition wird also verkleinert. Bei geringer bzw. sinkender Ressourcenauslastung können dem Experimentlauf aber auch mehr Knoten zur Verfügung gestellt werden, was einer Vergrößerung der Partition entspricht und eine höhere Parallelität ermöglicht.

Die jeweilige Partitionsgröße kann dabei durch zwei Grenzen beeinflusst werden: Zum einen eine systemweit festgelegte Obergrenze, so dass für Batch- und interaktive Experimente immer eine Mindestanzahl an Knoten verfügbar ist. Zum anderen kann die Priorität eines Experimentlaufs dazu verwendet werden, die Größe der Partition in Verhältnis zu allen verfügbaren Ressourcen festzulegen, um so Experimentläufe mit höherer Priorität einen höheren Parallelisierungsgrad zu ermöglichen.

Die Verwaltung der Partitionen kann dabei von der Experimentlauf-Steuerung übernommen werden. Dazu ist keine globale Ressourcenverwaltung nötig, da die Entscheidung über eine Änderung der Größe einer Partition aus der Auslastung zum Zeitpunkt der Aktivierung eines Teilexperiments getroffen werden kann.

## 8.2 Zusammenfassung

Das Ziel der Arbeit war, die Durchführung parametrisierbarer Experimentläufe in einem Emulationssystem für Rechnernetze zu unterstützen. Die physische Umgebung wurde dabei durch den im Rahmen des NET-Projekts an der Universität Stuttgart aufgebauten PC-Cluster bereitgestellt. Als Grundlage für die Unterstützung von parametrisierten Experimentläufen diente das an der University of Utah entwickelte Software-System Emulab. Das System bietet bereits Möglichkeiten, um einzelne Experimente durchführen zu können. Dazu werden von Emulab Dienste zur Benutzerverwaltung, Zugangskontrolle und Ressourcenverwaltung bereitgestellt, und eine Ablaufsteuerung übernimmt Aufgaben wie Auswahl, Bereitstellung und Konfiguration von Ressourcen. Durch einen Stapelbetrieb wird eine automatisierte Durchführung von Experimenten unterstützt, ein interaktiver Betrieb ermöglicht das Durchführen von Experimenten

unter direkter Einflussnahme eines Experimentators.

Das Emulab-System wurde für den Einsatz im existierenden Emulationssystem der Universität Stuttgart zunächst portiert. Emulab wurde dann erweitert, um eine Durchführung von parametrisierten Experimentläufen zu ermöglichen. Die Realisierung wurde durch die Integration in das Emulab-System so gestaltet, dass Benutzer einfach und effizient parametrisierte Experimentläufe durchführen können. Dazu wurde die bereits bestehende Funktionalität für einzelne Experimente herangezogen und auf parametrisierte Emulationsläufe abgebildet und erweitert.

Eine Ausführung von Experimenten unter verschiedenen Randbedingungen wird durch die Definition von Parametern und deren Wertemengen möglich. Daher wurde für Experimente eine Unterscheidung in Meta-Experimente und Teilexperimente getroffen. Alle Kombinationen der Parameterwerte werden über Teilexperimente im System abgebildet. Die Verwaltung und Durchführung aller Teilexperimente wird dann vom Benutzer durch Kommandos an das Meta-Experiment vorgenommen.

Aus einer parametrisierten Szenariobeschreibung werden beim Anlegen eines Experimentlaufs alle Parameterkombinationen abgeleitet, und daraus Szenariobeschreibungsinstanzen für alle Teilexperimente gebildet. Durch diese Methode können alle Komponenten einer Szenariobeschreibung einer Parametrisierung unterzogen werden. Dabei wird für Szenariobeschreibungen weiterhin die Skriptsprache oTcl mit Emulab-spezifischen Erweiterungen verwendet. Szenariobeschreibungen sind daher ohne Lernaufwand weiterhin flexibel und einfach zu definieren.

Die Steuerung eines parametrisierten Experimentlaufs erfolgt für alle Teilexperimente ganzheitlich durch das Meta-Experiment. Dazu werden von einem Benutzer Kommandos an das Meta-Experiment abgesetzt. Der Ablauf und die Koordination der Teilexperimente wird vom System durch die Experimentlauf-Steuerung geregelt, und die Durchführung der Teilexperimente findet automatisiert statt. Zusätzlich wurde eine Erfolgsverwaltung implementiert, die nicht nur ein Anhalten und Fortsetzen eines Experimentlaufs ermöglicht, sondern auch Auskunft über den Verlauf einer Durchführung gibt. Treten während einer Durchführung Fehler auf, so werden einem Benutzer durch Erweiterungen der Szenariobeschreibungssprache Möglichkeiten gegeben, das weitere Vorgehen zu bestimmen.

Da die Laufzeit von Experimentläufen gegenüber einzelnen Experimenten steigt, muss bei der Unterstützung von parametrisierten Experimentläufen eine effiziente Durchführung erreicht werden. Dazu wurden zwei Mechanismen eingeführt. Die Parallelisierung von Teilexperimenten verkürzt die gesamte Durchführungsdauer von Experimentläufen. Um ein faires Verhalten zwischen den bereits in Emulab existierenden Ausführungsarten und Experimentläufen zu erzielen, muss die theoretisch erreichbare Parallelität allerdings beschränkt werden. Der Parallelisierungsgrad wird dazu durch die maximale Anzahl reservierbarer Knoten bei einer Durchführung begrenzt. Zusätzlich wird durch eine Vorabreservierung der Knoten garantiert, dass ein erfolgreich aktivierter Experimentlauf durchgeführt werden kann, ohne aufgrund mangelnder Ressourcen unterbrochen werden zu müssen.

Als zweiter Mechanismus zur Verkürzung der Durchführungsdauer, wird eine Wiederverwendung von Knoten ermöglicht. Die Wiederverwendung von Knoten führt zu einem kurzen Zeitintervall zwischen Teilexperimenten bei deren Hintereinanderausführung. Anstatt Knoten zwischen zwei Teilexperimenten durch Aufspielen eines Festplattenabbildes neu zu initialisie-

ren, wird lediglich deren Konfiguration bei der Bereitstellung der Ausführungsumgebung angepasst. Dazu wurde die Knotenverwaltung von Emulab angepasst. Beide Mechanismen führen dazu, dass die gesamte Durchführungsdauer eines Experimentlaufs reduziert wird.

Um die Effizienz beider Mechanismen nachzuweisen, wurden mehrere Messungen vorgenommen. Dabei kamen zwei Szenarien zum Einsatz. Die Szenarien wurden einmal als parametrisierter Experimentlauf und einmal als einzelne Experimente im Stapelbetrieb durchgeführt. Damit wurde ein Vergleich der neu entwickelten Experimentlauf-Steuerung mit der in Emulab bereits existierenden Steuerung für den Stapelbetrieb möglich.

Die Messergebnisse zeigen, dass Experimentläufe durch die Parallelisierung von Teilexperimenten und die Wiederverwendung von Knoten durchgängig kürzere Laufzeiten aufweisen. Beide Mechanismen tragen dazu bei, gegenüber dem Stapelbetrieb die Laufzeiten bedeutend zu verkürzen und teilweise um die Hälfte zu reduzieren. Dabei ist die Ressourcenbelastung während der Durchführung von Teilexperimenten zwar höher als im Stapelbetrieb, durch eine kürzere Belegungszeit werden die Ressourcen aber insgesamt effizienter eingesetzt.

Parametrisierte Experimentläufe sind damit in Emulab auf integrierte, einfache und effiziente Weise möglich, und können einen Experimentator hilfreich bei Evaluationen unterstützen.

# Anhang A

# MANET Szenariobeschreibung

Listing A.1: oTcl-Skript des parametrisierten MANET Szenarios

```
1  set ns [new Simulator]
2  source tb_compat.tcl
3
4  tb-exprun-expfailure      ignore
5  tb-exprun-onreclaim       reconfigure
6
7  set exp_time              120
8
9  set nodes_list            [list 6 8 12]
10 set speed_list            [list 5 10 20]
11
12 # - - - - - - - -
13 proc make-topo {} { uplevel #0 {
14
15   # environment
16   set opt(EXPRUN_ID) MANET_ER_${this_nodes}Nodes_${this_speed}
17
18   # make nodes
19   set lan_members ""
20   for {set i 0} {$i < $this_nodes} {incr i} {
21       set node$i [$ns node]
22       if {$i > 0} {
23           tb-set-node-startcmd node$i "/bin/true"
24       }
25       append lan_members "node$i "
26   }
27
28   # set real start command
29   tb-set-node-startcmd node0 "~/scripts/MaNET-SUT.sh $exp_time $this_speed"
30
31   # make lan
```

```
32   set lan   [$ns make-lan $lan_members 1Gbps 0ms]
33
34 }}
35
36 # ---------------
37 foreach this_nodes $nodes_list {
38
39   foreach this_speed $speed_list {
40       make-topo
41       $ns run
42   }
43
44 }
```

# Literaturverzeichnis


[Baj98] Sandeep et al Bajaj. Virtual InterNetwork Testbed: Status and Research Agenda. Technical report, University of Southern California, July 1998.

[BDSY88] D. F. Bacon, A. Dupuy, J. Schwartz, and Y. Yemini. NEST: A network simulation and prototyping tool. In *USENIX Conference Proceedings*, pages 71–77, Dallas, TX, 1988.

[BMT+98] Rajive Bagrodia, Richard Meyer, Mineo Takai, Yu an Chen, Xiang Zeng, Jay Martin, and Ha Yoon Song. PARSEC: A Parallel Simulation Environment for Complex Systems. *IEEE Computer*, 31(10):77–85, October 1998.

[CEV03a] Roberto Canonico, Donato Emma, and Giorgio Ventre. An XML Based Network Simulation Description Language. In *European Simulation and Modelling Conference (ESMC2003)*, Naples, Italy, October 27–29 2003.

[CEV03b] Roberto Canonico, Donato Emma, and Giorgio Ventre. An XML description language for Web-based network simulation. In *Distributed Simulation and Real-Time Applications*, pages 76–81, Naples, Italy, October 23–25 2003.

[Fal99] Kevin Fall. Network Emulation in the Vint/NS Simulator. In *Proceedings of the Fourth IEEE Symposium on Computers and Communications (ISCC'99)*, July 1999.

[Fou03] Foundry Networks, Inc. *JetCore Base Chassis Systems - An Architecture Brief on NetIron BigIron, and FastIron Systems*, 2003.

[Fou04] Foundry Networks, Inc. *Foundry Switch and Router Installation and Basic Configuration Guide*, July 2004.

[Gro05a] The Flux Research Group. About project groups. http://www.emulab.net-/tutorial/docwrapper.php3?docname=groups.html, May 05 2005. Online Documentation.

[Gro05b] The Flux Research Group. Emulab Advanced Example. http://www.emulab.net-/tutorial/docwrapper.php3?docname=advanced.html, May 05 2005. Online Documentation.

[Gro05c] The Flux Research Group. Emulab Advanced Tutorial. http://www.emulab.net-tutorial/docwrapper.php3?docname=tutorial2.html, May 05 2005. Online Documentation.

[Gro05d] The Flux Research Group. Emulab Getting Started Tutorial. http://www.emulab.net/tutorial/docwrapper.php3?docname=tutorial.html, May 05 2005. Online Documentation.

[Gro05e] The Flux Research Group. Emulab-specific NS Extensions Reference Manual. http://www.emulab.net/tutorial/docwrapper.php3?docname=nscommands.html, May 05 2005. Online Documentation.

[Gro05f] The Flux Research Group. Emulab System Architecture – Condensed. http://www.emulab.net/doc/docwrapper.php3?docname=arch-small.html, May 05 2005. Online Documentation.

[Gro05g] The Flux Research Group. Node Usage Policies and Swapping Experiments. http://www.emulab.net/tutorial/docwrapper.php3?docname=swapping.html, May 05 2005. Online Documentation.

[Gro05h] The Flux Research Group. Testbed Master Control Daemon (TMCD) Reference Manual. http://www.emulab.net/tutorial/docwrapper.php3?docname=tmcd.html, May 05 2005. Online Documentation.

[GSHL03] Shashi Guruprasad, Leigh Stoller, Mike Hibler, and Jay Lepreau. Scaling Network Emulation with Multiplexed Virtual Resources. In *SIGCOMM 2003 Poster Abstract*, August 2003.

[HLR02] Daniel Herrscher, Alexander Leonhardi, and Kurt Rothermel. Modeling Computer Networks for Emulation. In *Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '02)*, pages 1725–1731, Las Vegas, June 2002.

[HM04] Daniel Herrscher and Steffen Maier. *NET: The Network Emulation Testbed Manual*. Institut für parallele und verteilte Systeme, Universität Stuttgart, July 2004.

[HR02] Daniel Herrscher and Kurt Rothermel. A Dynamic Network Scenario Emulation Tool. In *Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN 2002)*, pages 262–267, Miami, October 2002.

[HSL+03] Mike Hibler, Leigh Stoller, Jay Lepreau, Robert Ricci, and Chad Barb. Fast, scalable disk imaging with frisbee. In *Proc. of the 2003 USENIX Annual Technical Conf.*, pages 283–296, San Antonio, TX, June 2003. USENIX Association.

[Kes88] Srinivasan Keshav. Real: A network simulator. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1988.

[Kes89] Srinivasan Keshav. Real manuals. Technical Report UCB/CSD-89-530, EECS Department, University of California, Berkeley, 1989.

[Leu04] James R. Leu. Linux Virtual Routing and Forwarding, 2004.

[RAL03] Robert Ricci, Chris Alfred, and Jay Lepreau. A Solver for the Network Testbed Mapping Problem. In *ACM SIGCOMM Computer Communications Review*, volume 32, pages 65–81, April 2003.

[Riz97] Luigi Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, January 1997.

[ROLV06] Robert Ricci, David Oppenheimer, Jay Lepreau, and Amin Vahdat. Lessons from resource allocators for large-scale multiuser testbeds. *SIGOPS Oper. Syst. Rev.*, 40(1):25–32, 2006.

[Rot02] Jörg Roth. *Mobile Computing*. dpunkt Verlag, 2002.

[Sch00] Jochen Schiller. *Mobilkommunikation - Techniken für das allgegenwärtige Internet*. Addison-Wesley, 2000.

[Tan92] Andrew S. Tanenbaum. *Modern Operating Systems*. Internals and Design Principles. Prentice-Hall, International, 1992.

[Var98] András Varga. Parametrized Topologies for Simulation Programs. In *Proceedings of the Western Multiconference on Simulation (WMC'98) / Communication Networks and Distributed Systems (CNDS'98)*, San Diego, CA, January 11–14 1998.

[VP97] A. Varga and G. Pongor. Flexible Topology Description Language for Simulation Programs. In *Proceedings of the 9th European Simulation Symposium*, Passau, Germany, October 19–22 1997.

[Wer98] A. Werner. Linux traffic control — implementation overview. Technical Report SSC/1998/037, EPFL, November 1998. ftp://lrcftp.epfl.ch/pub/people/almesber/pub/tcio-current.ps.gz.

[WL95] David Wetherall and Christopher J. Linblad. Extending Tcl for Dynamic Object-Orientated Programming. In *Proceedings if the USENIX Tcl/Tk Workshop*, page 288, Toronto, Ontario, July 1995.

[WLS+02] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, pages 255–270, Boston, MA, December 2002. USENIX Association.