Sicherheitsaspekte bei Deployment virtueller Netzwerkinfrastrukturen

Gerhard Gröschl, Miran Mizani
{gerhard.groeschl, miran.mizani}@campus.lmu.de

Seminar: Trends in Mobilen und Verteilten Systemen
Wintersemester 2016/2017

Lehrstuhl für Mobile und Verteilte Systeme
Institut für Informatik
Ludwig-Maximilians-Universität München

Betreuer: Michael T. Beck
eingereicht am 6. Januar 2017

FEHLT [GG]: Optimierung der Grafiken, Erkl?ung der Ansatz2-Algorithmen, Ansatz2 Komplexit? und Performanz, Vergleich, Einf?bung der Gleichung Ansatz1, Legende maMo Ansatz2

FRAGEN[GG]: mathematische Modelle rausnehmen?

Abstract: Durch die ??rst effiziente Nutzung von Hardware mittels Virtualisierung steigt die Nachfrage nach virtualisierten Infrastrukturen enorm. Serviceprovider m?ssen die Hardwarebasis f?r ihre Dienste nicht mehr selbst unterhalten und geben ihre dahingehende Verantwortung an Infrastrukturanbieter weiter. Um die Sicherheit dieser Strukturen nicht zu vernachl?sigen, arbeiten viele Forscher in diesem Bereich und versuchen effiziente Algorithmen mit integrierter Beachtung der Sicherheitsaspekte zu finden. Diese Arbeit soll einen ?erblick und eine Klassifizierung der Gefahren, die solche Konstrukte betreffen, sowie eine Analyse zweier unterschiedlicher Ans?ze zur Vermeidung m?glichst vieler Risiken zum Zeitpunkt der Planung ?bermitteln.

Inhaltsverzeichnis

1 Einleitung

Ein Konzept dem Internet Impasse mit flexibler Architektur [WAS IST DAS?] und Handhabbarkeit zu begegnen, wurde in der Netzwerkvirtualisierung (NV) gefunden. [APST05, BOB+12, FBB+13] Sie basiert auf Knoten- (z.B. Xen) und Linkvirtualisierung [ANGABE ZU BEIDEM?] und erlaubt so von der tatsächlichen physischen Hardware beinahe unabhängige logische bzw. virtuelle Netzwerke einzurichten, welche nach außen hin den Anschein physischer Netzwerke erwecken. Die Möglichkeit mehrere virtuelle Maschinen (VMs) pro physischem Host und verschiedene heterogene virtuelle Netzwerke (VNs) auf demselben physischen Substratnetz zu betreiben befördert die Flexibilität der Netzwerkarchitektur und wirkt dem Internet Ossification Problem [APST05] entgegen.

Die großen Vorteile der NV liegen in der Abstraktion von der eingesetzten Hardware. Das Erstellen, Verändern, Migrieren, Zurücksetzen und Löschen von Maschinen funktioniert genauso einfach wie der Umgang mit Dateien, was eine dynamischere Nutzung des Netzwerkes erlaubt. Virtuelle Maschinen und Netzwerke eigenen sich auch als Testumgebung. Einerseits werden bestehende Systeme im Fehlerfall nicht direkt beeinträchtigt. Andererseits kann neuer Code nun leicht in verschiedenen Umgebungen (Windows, Linux, verschiedengroßer RAM, mit oder ohne Software-Developer-Kits etc.) ohne zusätzliche Hardware getestet und später einfach ausgerollt werden.

NV eröffnet eine Unterteilung des klassischen Internetserviceproviders (ISP) in Service-Provider (SP) und Infrastructure-Provider (InP). Damit gewonnene Freiheiten durch z.B. jeweils unabhängige Technologieentscheidungen[wang2016towards] sind wohl besonders für Unternehmen interessant, die die Hardwarebasis ihrer Dienste nicht mehr selbst unterhalten wollen.
Das Anbieten von Software und Hardware als on-demand Ressourcen wird wegen geringeren Wartungsaufwand, verminderte Hardwarekosten durch Koexistenz mehrerer Mieter, aber v.a. wegen Automatisierbarkeit in der Programmierung der Netzwerkumgebung vereinfacht. Dass InPs nicht mehr streng durch Hardware limitiert sind, begünstigt Skalierbarkeit und bspw. lastbedingte Migration von VMs auf andere physische Hosts.
Auch für den Kunden bietet NV Vorteile: Unternehmen bezahlen nur noch für diejenigen Ressourcen, die gerade in Anspruch genommen werden. Hochqualitative Hardware kann so zu einem Bruchteil ihres Preises erworben und ungenutzte Hardware reduziert werden. Durch dynamisches Skalieren (z.B. in Zeiten hoher Last) kann die eigene IT-Landschaft mühelos vergrößert werden.

Das Outsourcing von Rechenleistung, Speicher, Inhalten und Netzwerk soll Soft- und Hardware einfacher nutzbar machen und Geschäftsprozesse befördern. Die damit einhergehende Verantwortungsübertragung erfordert eine Anpassung des Risikomanagements und IT-Sicherheitstechnische Arbeiten zur Erhaltung der klassischen C.I.A.-Aspekte. Wegen der gemeinsam genutzten Hardware kommt aus Sicht des Kunden besonders der Isolation und dem Datenschutz eine wichtige Rolle zu.
Bekannte Sicherheitsmechanismen wie Verschlüsselung, Firewalls, Intrusion Detection Systeme etc. können zwar auf den virtuellen Komponenten des Netzwerks implementiert werden. Die Sicherheit von Nutzerdaten könne dadurch aber wegen der heterogenen und stark dynamischen Struktur virtueller Umgebungen jedoch nicht garantiert werden. Ferner

dürften Vorteile der Netzwerkvirtualisierung durch den zusätzlichen Overhead verloren gehen. [GCH+16]
Eine mögliche Lösung hierzu ist das Integrieren von Sicherheitsaspekten bereits in die Zuordnung von virtuellen zu physischen Knoten und Links. Dieser Virtual Network Embedding (VNE) Prozess stellt eine der größten Herausforderungen in der Netzwerkvirtualisierung dar. [FBB+13] Werden virtuelle Netzwerke entsprechend ihrer Sicherheitsanforderungen bereits auf Substratknoten mit hinreichender Schutzfunktion wie beispielsweise Firewall abgebildet, so kann Overhead durch zusätzliche Sicherheitstechnik im laufenden Betrieb des VNs reduziert werden.

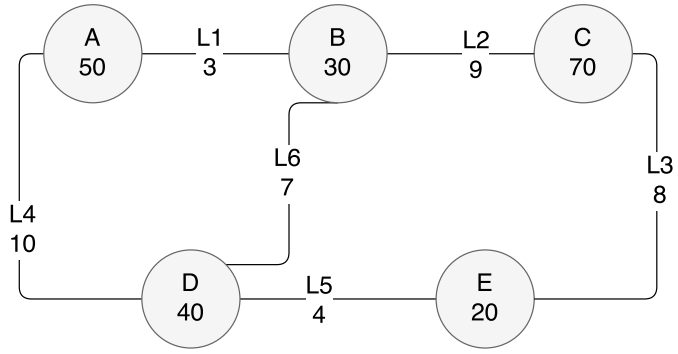
Derzeit wird versucht [BOB+12, GCH+16, WCC16] diese Probleme bereits im VNE-Algorithmus anzugehen (SVNE S

Diese Arbeit soll Gefahren im Kontext virtualisierter Netzwerke klassifizieren und zwei unterschiedliche Ansätze zur Schaffung eines möglichst hohen Sicherheitsniveaus bereits zum Zeitpunkt des VNE-Prozesses analysiert.
Dazu wird zuerst das VNE-Problem in Kapitel 2 *Das Virtual Network Embedding Problem* dargestellt. Kapitel 3 *Sicherheitsaspekte virtueller Netzinfrastrukturen* untersucht Sicherheitsanforderungen an virtuelle Netzwerkstrukturen und klassifiziert Sicherheitsrisiken, die sich in deren Kontext ergeben. Zwei Möglichkeiten zur Vermeidung von Gefahren, denen bereits im VNE-Prozess begegnet werden kann, werden im Kapitel 4 *Vermeidung von Gefahren via Secure VNE (SVNE)* betrachtet. Nach einer Diskussion in dieser Arbeit offengebliebener Probleme in Kapitel 5 *Ungelöste Probleme* wird mit einem Ausblick abgeschlossen.

2 Das Virtual Network Embedding Problem

Um den Anforderungen der heutigen Gefahren zu gen?gen, ist es unumg?glich s?tliche Grunds?ze der IT-Sicherheit m?glichst fr?h in die Planung der gew?nschten Infrastruktur miteinzubeziehen. Nicht nur, weil das nachtr?liche Schlie?n von Sicherheitsl?cken und Hinzuf?gen von Sicherheitskomponenten in finanzieller und zeitlicher Hinsicht zehnmal so teuer ist wie die initiale Beachtung dieser Aspekte, sondern weil die vollst?dige Sicherheit eines nachger?steten Systems kaum gew?rleistet werden kann [Col11].„Security-by-Design“ ist einer der wichtigsten Begriffe bei der Planung ?ffentlich zug?glicher Netzwerkstrukturen. Dementsprechend gro?ist die Nachfrage nach VNE-Algorithmen, die bereits beim Prozess des Mappings m?glichst viele Sicherheitsaspekte beachten und abdecken. Die vorausgehende Klassifizierung der bekannten Gefahren in „VNE-relevant“ und „nicht-VNE-relevant“ bildet die Grundlage f?r unsere weiteren Untersuchungen. Um zuvor noch einen genaueren Einblick in die Problematik von VNE zu gew?ren, beginnen wir zun?hst mit der Erl?terung des VNE-Problems sowie einem ?erblick ?ber die verschiedenen erfolgreichen Strategien bestehender VNE-Algorithmen, welche die Sicherheitsaspekte noch nicht beachten.

Betrachtet man das zugrundeliegende physische System als einen ungerichteten Graphen



und extrahiert die vorhandenen Elemente sowie deren Werte, erh?t man die Basismenge

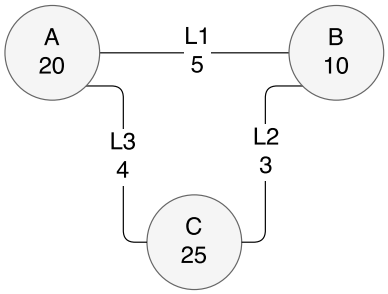
$$G^S = \{N^S, L^S\}$$

(N steht f?r die Menge der physischen Knoten, L f?r die Menge der physischen Links)

$$G^S = \{\{ (A^S,50),(B^S,30),(C^S,70), (D^S,40),(E^S,20)\},$$
$$\{ (L1^S,3),(L2^S,9),(L3^S,8), (L4^S,10),(L5^S,4),(L6^S,7)\}\}$$

in welche die virtuellen Strukturen eingebettet werden sollen.

Ein „virtual network request“ (in Folge „VNR“ genannt) kann hierbei ebenfalls als ein ungerichteter Graph gesehen



und genauso in eine Menge ?bersetzt werden.

$$G^V = \{N^V, L^V\}$$

(N steht f?r die Menge der virtuellen Knoten, L f?r die Menge der virtuellen Links)

$$G^V = \{\{ (A^V,20),(B^V,10),(C^V,25)\} , \{ (L1^V,5),(L2^V,3),(L3^V,4)\}\}$$

Gesucht ist nun eine Abbildungsfunktion $f : G^V \rightarrow G^S$

In der Regel ist es ?blich, nicht nur eine einzelne VNRs auf ein physisches System abzubilden, sondern gleich mehrere auf einmal. Ebenso w?e es theoretisch m?glich, dass ein Infrastruktur-Provider gleich mehrere getrennte unabh?gige physische Strukturen betreibt, und somit mittels VNE das beste System f?r die Abbildung der Menge von VNRs eroiren m?chte. Dies k?nnte beispielsweise der Fall sein, wenn ein Serviceprovider ein VNR mit der Bedingung „alle Elemente m?gen sich am selben Standort befinden, egal an welchem“ in Auftrag gibt, und der Infrastruktur-Provider ?ber mehrere Hardware-Standorte verf?gt. Die Attributmenge bei den grundlegenden VNE-Algorithmen beschr?kt sich zumeist auf Rechnerleistung und Bandbreite. Lokalit?, GPU-Leistung und RAM-Menge w?en einige weitere m?gliche Standard-Attribute. Ein gro?s Problem bei der Berechnung des optimalen Mappings findet sich bei der Berechnungszeit. Die endliche Beschr?kung der Knoten- sowie Link-Resourcen und die „on-line nature“ von VNRs stellen zus?zliche Hindernisse dar. Da selbst L?sungen f?r einfache Anfragen (geringe Anzahl von Knoten und Links), welche wenige Attribute beinhalten, exponentiellen Rechenaufwand ben?tigen, steigt der Aufwand sowohl mit der Anzahl der abzubildenden Knoten und Links, als auch mit steigender Attributanzahl dementsprechend. Teilweise gilt das Problem als rechnerisch unl?sbar, grunds?zlich aber befinden wir uns im Komplexit?sbereich der NP-Vollst?digkeit [MRR13]. Da die Erh?hung der Attributmenge - wie bereits genannt - grunds?zlich nicht positiv zur Laufzeit der existierenden Algorithmen beitr?t, werden wir den Komplexit?sbereich beim Hinzuf?gen von Sicherheitsanforderungen nicht verlassen. Dennoch gibt es M?glichkeiten, die den Rechenaufwand reduzieren k?nnen. Im Folgenden widmen wir uns allerdings zuerst einer ?ersicht ?ber die Gefahren, welche bei VNE eine Rolle spielen.

3 Sicherheitsaspekte virtueller Netzinfrastrukturen

3.1 Sicherheitsanforderungen

Dummytext. :)

3.2 Herkömmliche Gefahren in Netzinfrastrukturen

3.3 Neue Verwundbarkeiten in virtualisierten Umgebungen

Dummytext. :)

Technischer Art

Dummytext. :)

Organisatorischer Art

Dummytext. :)

Rechtlicher Art

Dummytext. :)

3.4 VNE-Relevante Gefahren

4 Vermeidung von Gefahren via Secure VNE (SVNE)

Da wir nun einen ?erblick zu VNE sowie den bestehenden Gefahren gegeben haben, widmen wir uns nun zwei verschiedenen SVNE-L?sungsans?zen.

4.1 Ansatz 1

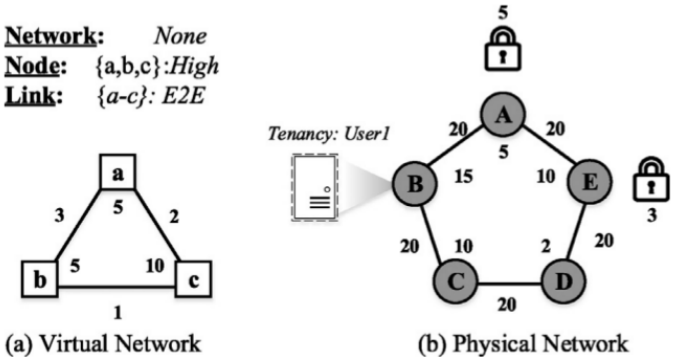
Hierbei besch?tigen wir uns mit dem L?sungsansatz aus [WCC16]. Das Hauptaugenmerk bez?glich der Sicherheitsaspekte legen Wang et al auf Traffic-Verschl?sselung und die Separierung von VMs unterschiedlicher Trust-Levels. Mittels„Security plan design“werden die drei folgenden strukturellen Aspekte betrachtet und in dementsprechende Levels eingeteilt.

- The Network plan:
  Ein VNR wird als Netzwerk betrachtet und seinem Level entsprechend isoliert. „High“fordert und beansprucht ein gesamtes Netz bzw. Subnetz der physischen Infrastruktur f?r sich. Hiermit soll die Wahrscheinlichkeit f?r DOS-Angriffe oder ?nliche vermindert werden, da„Mehr-Parteien-Netzwerke“die Hauptschwachstellen f?r solche Angriffe darstellen [Liu10]. Auch Sniffing durch kompromittierte Hosts im selben Netz wird durch diese Ma?ahme verhindert. W?rend ”high“Resourcenteilung somit komplett verweigert, l?st der Level„medium“zumindest ein gemeinsam genutztes Netz f?r VNRs vom selben Eigent?mer zu.
- The Node plan:
  Die einzelnen virtuellen Knoten eines VNR stellen Isolierungsanforderungen an die physischen Knoten. Wie auch beim„network plan”, werden die Levels„high”„medium“und„none“definiert und umgesetzt.„high“fordert die alleinige Existenz eines

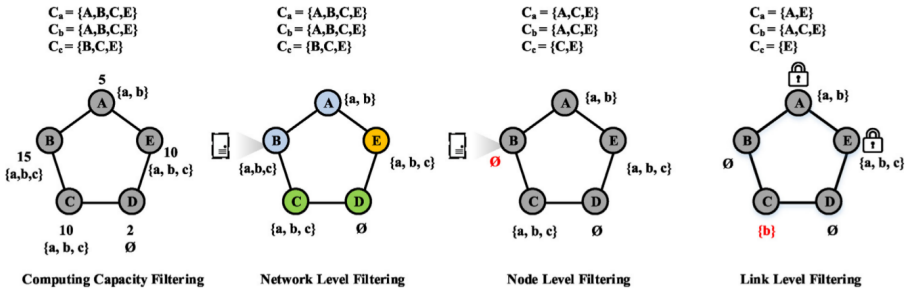
VNR-Knotens„medium“l?st VNR-Knoten vom selben Eigent?mer auf ein und demselben physischen Knoten zu. Durch diesen Plan sollen Angriffe von VM zu VM ?ber gemeinsam genutzte Resourcen unterbunden werden. Zus?zlich wird das Risiko eines Angriffs vom physischen Host verringert, da der Angriffsvektor„VM zu physischem Host“eliminiert wird.

- The Link plan:
  End-to-End(E2E), Point-to-Point(P2P) und„none“sind die hier w?lbaren Levels. W?rend E2E nur an den Endpunkten Verschl?sselungskapazit?en zu Verf?gung stellen muss, ben?tigt P2P diese Kapazit?en an allen Hops des abgebildeten Links. Die heutzutage sehr g?gigen man-in-the-middle Angriffe sollen dadurch wesentlich erschwert werden.

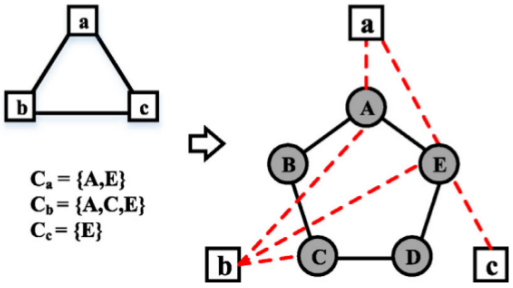
VNRs werden nun, wie in Kapitel 2 bereits gezeigt, in ungerichtete Graphen mit Standardanforderungentransformiert. Zus?zlich werden hier auch noch Sicherheitsanforderungen integriert.



Nun wird ein vier-stufiges Pre-Processing durchgef?hrt, um die Berechnung der optimalen Abbildung der VNRs zu vereinfachen. Hierbei werden als ersten die Standardanforderungen der virtuellen Knoten mit den zu Verf?gung stehenden Kapazit?en der physischen Knoten verglichen. Sollten physische Knoten bestimmte Anforderungen nicht erf?llen, werden sie aus der Berechung ausgeschlossen. Der zweite Schritt ist dem„Network plan“gewidmet. Sollte ein physischer Knoten, welcher den Anforderungen des„Network plan“nicht gen?gt sich in einem Kandidaten-Netzwerk befinden, wird das Subnetz aus den Berechnungen ausgeschlossen. Im dritten Schritt werden die Sicherheitsanforderungen der einzelnen Knoten betrachtet und nicht entsprechende physischen Knoten werden abermals entfernt. Der letzte Schritt widmet sich den Sicherheitsanforderungen aus dem„Link plan“und streicht Links aus den weiteren Berrechnungen, welche den Verschl?sselungsanforderungen der virtuellen Links nicht gen?ge tun.



Mit den aus dem Pre-Processing gewonnen Information bildet man nun einen Hilsgraphen, welcher die ?brig gebliebenen M?glichkeiten der Einbettung zeigt.



Dieses Pre-Processing reduziert das SVNE-Problem auf ein„multi-commodity-flow“Problem [mit einem Wert (Rohstoff?) pro Link]. [RA93] Im weiteren Vorgehen werden nun zwei F?le unterschieden.„Path-splitting”, im weiteren SVNE-PS und„no-path-splitting”, im weitern SVNE-NPS, sind zus?zliche Sicherheitsvorgaben, welche im Vorfeld definiert werden m?ssen, um die Wahl des Algorithmus zu erm?glichen. Sollte SVNE-PS gew?lt werden, beziehen sich die Gleichungen nur auf die Erf?llung der geforderten Attribute. Sollte SVNE-NPS gew?lt werden, werden die f?r Link-Abbildungen verantwortlichen Gleichungen ersetzt.

$$Min \sum_{p \in \psi} sf_p * ct_p + \sum_{(V,s) \in AL} cn_s * E_{V,s} * cr_V + ct_L \tag{1}$$

$$\sum_{V:(V,s) \in AL} E_{V,s} \leq 1, \quad \forall s \in N^S \tag{2}$$

$$\sum_{s:(V,s) \in AL} E_{V,s} = 1, \quad \forall V \in N^V \tag{3}$$

$$\sum_{p:x \in p} sf_p \leq \beta c_x, \quad \forall x \in L^S \tag{4}$$

$$\sum_{p \in \psi^c} sf_p = \beta r_c, \quad \forall c \in L^V \tag{5}$$

$$\sum_{p \in \psi_s^c} sf_p = \beta r_c, \quad \forall c \in L^V \tag{6}$$

$$\sum_{p \in \psi} \alpha_{p,(V,s)} * sf_p \leq E_{V,s} * T_V, \quad \forall (V,s) \in AL, V \in N^V, s \in N_S \tag{7}$$

$$sf_p \leq x_p * \beta r_c \quad \forall p \in \psi_s^c, c \in L^V \tag{8}$$

$$\sum_{p:x \in p, p \in \psi^c} x_p * \beta r_c \leq \beta c_x, \quad \forall x \in L^S \tag{9}$$

$$\sum_{p \in \psi^c} x_p = 1 \quad \forall c \in L^V \tag{10}$$

$$\sum_{p \in \psi_s^c} x_p = 1 \quad \forall c \in L^V \tag{11}$$

$$\sum_{p \in \psi} \alpha_{p,(V,s)} * x_p \leq E_{V,s} * D_V \quad \forall (V,s) \in AL, V \in N^V, s \in N^S \tag{12}$$

$$\sum_{p \in \psi} x_p * \theta_{p,i} \leq nc_i \quad \forall i \in N^S \tag{13}$$

ct_p: The unit cost of flow on path p, $ct_p \geq 0$;
H_p: The hop number of path p, $H_p \geq 1$;
cn_s: The unit cost of computing resources at physical node s, $cn_s \geq 0$;
ψ The set of all the paths for the model;
ψ^c The set of paths for virtual link c;
ψ_s^c The set of paths for virtual link c that consists of physical nodes all with cryptography capability;
cr_V The computing capacity requirement of each virtual node V;
βr_c The bandwidth requirement of virtual link c;
βc_x: The bandwidth capacity of physical link x;
nc_i: The number of cryptography instances that physical node i can support;
$\alpha_{p,(V,s)}$ 1 if auxiliary link (V, s) is on the path p, and 0 otherwise;
$\theta_{p,i}$ 1 if node i is contained in path p, and 0 otherwise;
ct_L: The cost of selected link security plan;
ct_L^c: The cost of selected link security plan for virtual link c;
ct_{ED}: The cost per cryptography instance;
AL: The set of auxiliary links;
D_V: The node degree of virtual node V in the virtual network;
T_V: The summation of the bandwidth requirements of all incident virtual links of virtual node V.

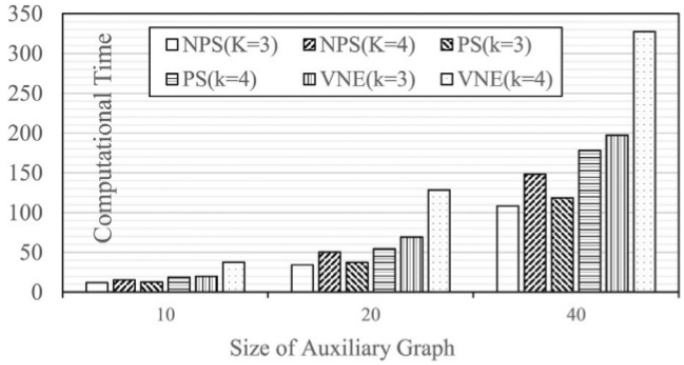
$E_{V,s}$: 1 if virtual node V is embedded to physical node s, and 0 otherwise;
sf_p: The size of the flow on path p, $sf_p \geq 0$;
x_p: 1 if there is a non-zero flow on path p, 0 otherwise.

Fehlt: Kurze Beschreibung der Punkte 1-13

Die Berechnungen wurden mittels induktiver logischer Programmierung unter Verwendung von CPLEX[hi] und durch k-shortest-path anhand aller abzubildenden Elemente limitiert. F?r die Testumgebungen wurden zuf?lig erzeugte VNRs mit 2 bis 10 Knoten und halbsovielen Links erzeugt. Auch die Rechenkapazit? wie Bandbreite wurden durch Zufallsgeneratoren mit Werten im Bereich von 1 bis 10 gew?lt und verteilt. Die Anzahl der VNRs betr?t einer Poisson-Verteilung nach einen Durchschnitt von 4 VNR pro 100 Zeiteinheiten, mit einer jeweiligen durchschnittlichen Einsatzzeit von 1000 Zeiteinheiten.

Die zugrundeliegenden physischen Netze wurden ebenfalls zuf?lig erzeugt und beinhalteten 10 bis 50 Knoten, sowie halbsoviele Links. Die Rechen- sowie Bandbreiten-Kapazit?en wurden gleichm?ig verteilt und enthielten Werte zwischen 1 und 50. Die Verschl?sselungskapazit?en der Knoten wurde ?ber die gesamte Testreihe ebenfalls gleichm?ig verteilt.

Die folgende Statisk zeigt einen Durchschnittsvergleich zwischen dem, als Standard gew?lten, VNE-Algorithmus[QH13] und den beiden SVNE-Varianten PS und NPS. Dazu sei gesagt dass s?tliche Algorithmen ihre Berechnungen anhand des Hilfsgraphen durchgef?hrt haben und durch k=3 bzw. k=4 limitiert wurden.



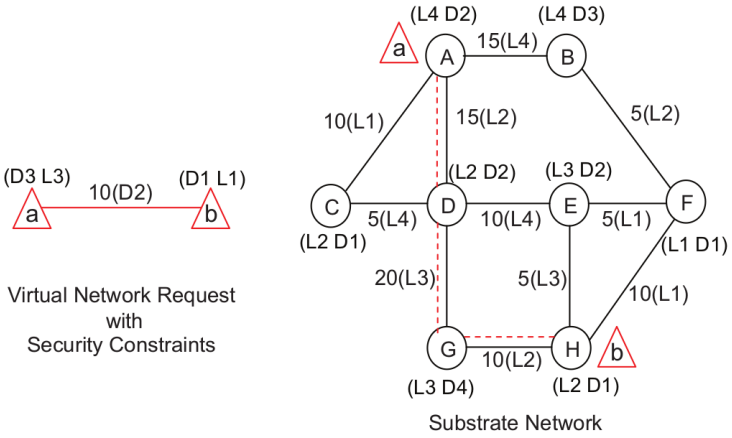
Man sieht hier einen deutlichen Zeitvorsprung von PS und NPS gegen?ber dem Standard-Algorithmus. Da hier jedoch auch der Standard-Algorithmus vom Pre-Processing profitiert, w?e ein Vergleich, welcher die Dauer des Pre-Processings mit aufnimmt und den Standard-Algorithmus ohne Pre-Processing arbeiten lie?, anschaulicher und aussagekr?tiger. Dennoch ist ein Punkt durchaus ?berraschend und bemerkenswert: Der Standard-Algorithmus ben?tigt deutlich l?ger, obwohl er keine Sicherheitsaspekte mitbeurteilt, im Gegenzug zu seinen Kontrahenten.

In welchem Komplexit?sbereich sich das Pre-Processing befindet wird hier leider nicht erw?nt.

4.2 Ansatz 2

Im zweiten Ansatz widmen wir uns dem Modell aus [SL15]. Dieser Ansatz arbeitet mit abstrakten Sicherheitslevels. Im Folgenden werden zwar Skalare hierf?r verwendet, was allerdings nicht zwingend so vorgesehen ist. Stattdessen w?e eine Verwendung komplexerer Sicherheitsvektoren m?glich, welche bei weitem detailliertere Sicherheitsmerkmale beschreiben k?nnten. Des weiteren verf?gt jeder physische wie auch virtuelle Knoten ?ber zwei Sicherheitswerte: Anforderungslevel und (eigenes) Sicherheitslevel. Der Anforderungslevel definiert das Minimum-Level des Gegen?bers, der Sicherheitslevel definiert die eigenen gew?nschten Sicherheitsmerkmale. Virtuelle Links verf?gen ?ber Anforderungslevels, physische Links nur ?ber Sicherheitslevels. Die vorausgesetzten Basisanforderungen, welchen alle VNRs unterliegen, beschr?ken sich auf 4 Regeln, welche mittels der zuvor genannten Merkmale umgesetzt werden:

- Ein physischer Knoten sollte einen Sicherheitslevel garantieren, der h?her ist als die Anforderungen der darauf abzubildenden virtuellen Knoten.
- Der Sicherheitslevel des virtuellen Knotens sollte h?her sein, als das Anforderungslevel des physichen Knotens.
- Alle virtuellen Knoten, welche auf den selben physischen Knoten abgebildet werden, sollten ?ber einen ausreichenden Sicherheitslevel verf?gen.
- Der Anforderungslevel des virtuellen Links sollte stets niedriger sein, als das Sicherheitslevel des physischen Links.



Wie man in Abbildung X sieht, werden auch bei diesem Modell VNRs und physische Netze in ungerichtete Graphen transformiert. Die in Klammern gestellten Parameter beschreiben Anforderungslevel(D) und Sicherheitslevel(L). Die hier durchgef?hrte Abbildung ber?cksichtigt nicht nur die Sicherheitsanforderungen aller Parteien unter Ber?cksichtigung der Basisanforderungen, sondern achtet zus?zlich noch auf Kostenminimierung. Insgesamt existieren drei Abbildungsm?glichkeiten, die gew?lte ist jedoch die g?nstigste, unter dem Aspekt keine Sicherheitsresourcen zu verschwenden. Dieser VNR h?te auch auf EDGH abgebildet werden k?nnen, wobei der Link ED ?ber einen Sicherheitslevel verf?gt, welcher h?her als n?tig ist. W?rde dies nicht beachtet werden, k?nnte es zu unn?tigen Engp?sen bei der Behandlung von VNRs mit h?heren Anforderungslevels kommen. Um ein solches Verhalten zu erreichen, wurden zus?zlich Kosten- und Nutzen-Funktionen in die Berechnung integriert.

$$\rho(i) = \begin{cases} 1, & \text{if request No. } i \text{ is accepted.} \\ 0, & \text{if request No. } i \text{ is denied.} \end{cases}, \forall i \in \{1, 2, \ldots, |M|\}. \tag{1}$$

$$\begin{aligned} Rev(M_i) = \rho(i) \times Dur_i^V \times \Bigg[& \sum_{n_i^V \in N_i^V} dem^V(n_i^V) \times cpu^V(n_i^V) \\ & + \sum_{l_i^V \in L_i^V} dem^V(l_i^V) \times bw^V(l_i^V) \Bigg]. \end{aligned} \tag{2}$$

$$\begin{aligned} Cost(M_i) = \rho(i) \times Dur_i^V \times \Bigg[& \sum_{n_i^V \in N_i^V} lev^S(M_{i,N}(n_i^V)) \times cpu^V(n_i^V) \\ & + \sum_{l_i^V \in L_i^V} lev^S(M_{i,L}(l_i^V)) \times len(M_{i,L}(l_i^V)) \times bw^V(l_i^V) \Bigg]. \end{aligned} \tag{3}$$

$$\max \quad \lim_{n \to \infty} \frac{\sum_{i=1}^{n} \rho(i)}{n}, where \rho(i) = \begin{cases} 1, & \text{if } VNR_i \text{ accepted.} \\ 0, & \text{if } VNR_i \text{ denied.} \end{cases}, \tag{4}$$

$$\max \quad \lim_{T \to \infty} \frac{\sum_{i=1}^{|\mathbf{M}|} Rev(M_i)}{T}, \tag{5}$$

$$\max \quad \lim_{T \to \infty} \frac{Rev(\mathbf{M})}{T} / \lim_{T \to \infty} \frac{Cost(\mathbf{M})}{T}. \tag{6}$$

$$\sum_{r=1}^{|N^S|} x_{i,qr} = 1, \qquad \forall n_{i,q} \in N_i^V, \tag{7}$$

$$\sum_{r=1}^{|P^S|} y_{i,qr} = 1, \qquad \forall l_{i,q} \in L_i^V, \tag{8}$$

$$\sum_{i=1}^{|N|} x_{i,qr} cpu_i^V(n_{i,q}) \leq cpu^S(n_r), \quad \forall n_{i,q} \in N_i^V, n_r \in N^S, \tag{9}$$

$$\sum_{i=1}^{|N|} y_{i,qr} bw_i^V(l_{i,q}) \leq \min_{l_j \in p_r} bw^S(l_j), \quad \forall l_{i,q} \in L_i^V, p_r \in P^S, \tag{10}$$

$$x_{qr} dem^S(n_r) \leq lev^V(n_q), \quad \forall n_q \in N_i^V, n_r \in N^S, \tag{11}$$

$$x_{qr} dem^V(n_q) \leq lev^S(n_r), \quad \forall n_q \in N_i^V, n_r \in N^S, \tag{12}$$

$$\begin{aligned} & \max\{dem^S(n_r), \max_{x_{qr}=1} dem^V(n_q)\} \\ & \leq \min\{lev^S(n_r), \min_{x_{qr}=1} lev^V(n_q)\} \end{aligned}, \quad \forall n_q \in N_i^V, n_r \in N^S, \tag{13}$$

$$dem^V(l_q) \leq \min_{l_i \in p_r, y_{qr} > 0} lev^S(l_i), \quad \forall l_q \in L_i^V, p_r \in P^S. \tag{14}$$

$$H^{(t+1)}(n,k) = \lambda \sum_{(m,n) \in Link(n)} PC((m,n),k) H^{(t)}(m,k) + (1-\lambda) H^{(t)}(n,k), \tag{16}$$

Fehlt: Kurze Beschreibung der Punkte 1-16

Das außergewöhnliche bei diesem Ansatz ist die Verwendung zweier unterschiedlicher Algorithmen, welche sich gegenseitig ergänzen: uSAV und cSAV.

- uSAV
  uSAV, der unkooridinierte zwei-Phasen-Algorithmus, behandelt Knoten- und Link-Abbildungen getrennt voneinander. Dementsprechend liegt die Schwäche von uSAV in der Abbildung von non-splittable-links. uSAV ist ein terminierender Algorithmus, welcher die Abbildung der Knoten priorisiert und dahingehend ein sehr gutes Ergebnis mit wenig Aufwand erzielt. Jedoch kann es, durch die Priorisierung der Knoten, zu hohen Kosten bei der Link-Abbildung kommen.

- cSAV
  cSAV, der koordinierte Algorithmus, behandelt Knoten und Links gemeinsam, und kann zur Optimierung, des bereits von uSAV gelieferten Ergebnisses verwendet werden. cSAV liefert zwar optimalere Ergebnisse als uSAV, benötigt aber auch mehr Zeit. Hier muss Zeitaufwand mit gelieferter Leistung abgewogen werden.

Beide Varianten nutzen die selben Heuristiken ((und Sub-Algorithmen)). Der Author schlägt für ideale Ergebnisse eine kombinierte Benutzung der beiden Varianten vor.

uSAV

1: For all requests in the time window, set the ones that did not expire yet but classified with `MAP_FAILED` to `NEW`.
2: Release the substrate resources that were occupied by `DONE` requests. Refresh the redundant network.
3: **repeat**
4: Get G_i^V that has the maximum revenue from all `NEW` requests in the time window.
5: Map the nodes of G_i^V using the node mapping algorithm.
6: **if** `MAP_NODE_SUCCESS` **then**
7: Map the links of G_i^V using the link mapping algorithm.
8: **if** `MAP_SUCCESS` **then**
9: Occupy the substrate resources. Refresh the redundant network.
10: Set the state of G_i^V to `MAP_SUCCESS`.
11: **return**
12: Set the state of G_i^V to `MAP_FAILED`.
13: Release the occupied resources of G_i^V.
14: **until** There is no more `NEW` requests in the time window
15: **return**

cSAV

Input: The redundant substrate network as G^S. The current interested virtual network request as G^V.

1: Initialize variables: N for maximum back-off times; two empty stacks S (for mapped nodes) and Q (for unmapped nodes).
2: Apply the heuristic computation in the virtual network. Find the virtual node m with the highest heuristic value $H(m)$. Let node $m' = m$.
3: **repeat**
4: Try to map m to $n \in G_S$, starting from substrate node with higher heuristic value.
5: **if** No applicable substrate node n to host virtual node m and corresponding virtual link mm' **then**
6: **if** $N == 0$ **then**
7: **return** `MAP_FAILED`
8: **else**
9: $N \leftarrow N - 1$
10: $Q.push(m)$, $m = S.pop()$.
11: $S.push(m)$.
12: Update the redundant network, recalculating heuristics.
13: **for all** Virtual node $m' \in Adj(m)$ in the decreasing order of $H(m')$ **do**
14: $Q.push(m')$
15: $m' = m$. $m = Q.pop()$.
16: **until** $Empty(Q) ==$ **true**
17: **return** `MAP_SUCCESS`

Sub-Algorithmus 1: Knoten-Abbildungen

1: For each possible security demand k, sort all substrate nodes in a candidate node queue $queue(k)$ in descending order of heuristic value H.
2: For all nodes $m \in G_i^S$, initialize their state by setting $Occupied(m) =$ FALSE.
3: **repeat**
4: Get an unmapped node n randomly from G_i^V.
5: $k = dem^V(n)$.
6: **if** $\exists$ node $m \in queue(k)$ s.t. $Occupied(m) =$ FALSE **and** $dem^S(m) \leq lev^V(n)$ **and** $cpu^S(m) \geq cpu^V(n)$ **then**
7: $Occupied(m) =$ TRUE.
8: Map the virtual node n onto the substrate node m.
9: **else**
10: Release all resources occupied by G_i^V.
11: **return** `MAP_FAILED`.
12: **until** all nodes in G_i^V are mapped successfully.
13: **return** `NODE_MAP_SUCCESS`.

Sub-Algorithmus 2: Link-Abbildungen

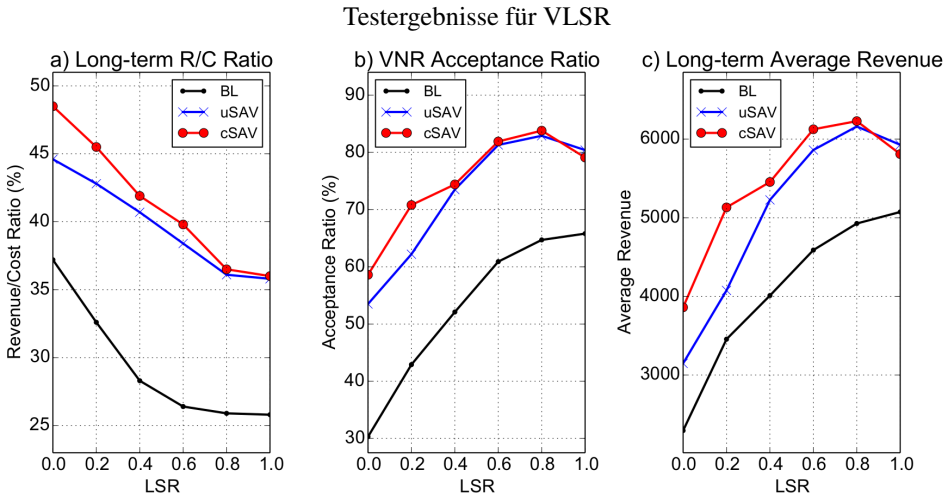
1: **for** each unmapped virtual link $l \in G_i^V$ **do**
2: $bw_r = bw^V(l), k = lev^V(l), flag = 0.$
3: **if** l is splittable **then**
4: $split = 1.$
5: **else**
6: $split =$ `MAX_SPLIT_TIME`.
7: Let $m_1, m_2 \in G^S$ be the hosts of both ends of l.
8: **repeat**
9: **if** $\exists p \in P^S$ s.t. $p : m_1 \rightarrow m_2$ has the minimum $PCC(p, k)$ **then**
10: $bw^S(p) = \min_{t \in p} bw^S(t), t \in L^S.$
11: Map the remaining resources of l onto p.
12: $bw_r = bw_r - bw^S(p).$
13: **if** $bw_r \geq 0$ **then**
14: $flag = 1.$
15: **else**
16: $split =$ `MAX_SPLIT_TIME` $+ 1.$
17: **until** $flag = 1$ **or** $split >$ `MAX_SPLIT_TIME`
18: **if** $flag = 0$ **then**
19: **return** `MAP_FAILED`.
20: **return** `MAP_SUCCESS`.

Die Testumgebung dieses Ansatzes wurde mittels GT-ITM-Tool cite-missing! erstellt.

Die physischen Netze wurden in der Größenordnung eines mittleres ISP angesetzt, und betrugen 100 Knoten und 500 Links. Die Bandbreite und Rechenkapazität wurden, wie auch im vorherigen Ansatz, gleichmäßig verteilt und betrugen Werte zwischen 50 und 100. Die abstrakten Sicherheitslevels der physischen Elemente wurden zwischen 0 und 4 gleichmäßig verteilt. Die Anforderungslevels wurden ebenfalls aus dem Bereich 0-4 gewählt, und so verteilt, dass kein physisches Element ein höheres Anforderungslevel als Sicherheitslevel besitzt.

Die VNRs beinhalteten 2 bis 20 Knoten sowie halbsoviele Links. Die Bandbreite- und Rechenkapazitäts-Forderungen wurden zwischen 0 und 50 gewählt und ebenfalls gleichmäßig verteilt. Die zeitlichen Parameter wurden auf durchschnittlich (Poisson-Verteilung) 5 Anforderungen pro 100 Zeiteinheiten begrenzt. Die Verwendungszeit eines VNRs folgt einer Exponential-Verteilung und betrugt im Durchschnitt 500 Zeiteinheiten. Eine einzelne Simulation erhielt 1500 VNRs und dauerte 30000 Zeiteinheiten.

Als Vergleichswert wurde der Algorithmus aus cite-6-missing verwendet. Die Tests wurden für drei, anhand der Link-Split-Ratio unterschiedenen Szenarien ausgelegt: High-LSR(mehr als 80% der Links dürfen gesplittet werden), Low-LSR(weniger als 20%) und VLSR(varied LSR). Da letzteres Testszenario am meisten Bezug zur Realität hat, werden wir hier nur diese Ergebnisse vorstellen.



Man sieht hier die drei, am schwersten gewichteten Aspekte dieses Ansatzes. Der Kosten-/Nutzen-Wert sinkt mit zunehmender LSR, während die Akzeptanz und der Nutzen gegenteiliges Verhalten zeigen. Die vorhandenen Laufzeitanalysen der Algorithmen bestätigen die Aussagen der Authoren. uSAV führt hier mit Werten zwischen 6 und 10 Minuten, während cSAV erst nach 14 bis 25 Minuten brauchbare Ergebnisse für einzelne VNRs liefert.

4.3 Vergleich

Um nun die beiden vorgestellten Ansätze in einen Vergleich zu bringen, möchten wir uns hier nicht auf die Ergebnisse der Testszenarien konzentrieren, sondern auf die unterschiedlichen Herangehensweisen.

5 Ungelöste Probleme

6 Schlussfolgerung und Ausblick

Frage: Auf welcher Ebene wird virtualisiert? Auf IP-Ebene? Was ist dann aber mit IP-Support-Protokollen (ARP)? Oder nicht-IP-Protokollen? Will ich ein Netzwerk virtualisieren, oder nur den IP-Verkehr? Verkapselung f?hrt zu Leistungseinbu?n. [CDRS07]

Literatur

[APST05] Thomas Anderson, Larry Peterson, Scott Shenker und Jonathan Turner. Overcoming the Internet impasse through virtualization. *Computer*, 38(4):34–41, 2005.

[BOB+12] Leonardo Richter Bays, Rodrigo Ruas Oliveira, Luciana Salete Buriol, Marinho Pilla Barcellos und Luciano Paschoal Gaspary. Security-aware optimal resource allocation for virtual network embedding. In *Proceedings of the 8th International Conference on Network and Service Management*, Seiten 378–384. International Federation for Information Processing, 2012.

[CDRS07] Serdar Cabuk, Chris I Dalton, HariGovind Ramasamy und Matthias Schunter. Towards automated provisioning of secure virtualized networks. In *Proceedings of the 14th ACM conference on Computer and communications security*, Seiten 235–245. ACM, 2007.

[Col11] Eric Cole. Network Security Bible: Edition 2, 2011.

[FBB+13] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer und Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906, 2013.

[GCH+16] Shuiqing Gong, Jing Chen, Conghui Huang, Qingchao Zhu und Siyi Zhao. Virtual Network Embedding through Security Risk Awareness and Optimization. *KSII Transactions on Internet & Information Systems*, 10(7), 2016.

[hi] http://www 01.ibm.com/software. CPLEX.

[Liu10] H. Liu. A new form of dos attack in a cloud and its avoidance mechanism, in: Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop, 2010.

[MRR13] Raouf Boutaba Muntasir Raihan Rahman. Surviveable Virtual Network Embedding Algorithms for Network Virtualization, 2013.

[QH13] X. Cao Q. Hu, Y. Wan. Resolve the virtual network embedding problem: a column generation approach. *INFOCOM, 2013 Proceedings IEEE*, 2013.

[RA93] J.B. Orlin R.K. Ahuja, R.L. Magnanti. Network Flow: Theory, Algorithms and Applications, 1993.

[SL15] Hong Xu Ming Xu Shuhao Liu, Zhiping Cai. Towards Security-aware Virtual Network Embedding, 2015.

[WCC16] Yang Wang, Phanvu Chau und Fuyu Chen. Towards a secured network virtualization. *Computer Networks*, 104:55–65, 2016.