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 3. Januar 2017

FEHLT [GG]: Optimierung der Grafiken, Erklärung der Ansatz2-Algorithmen, Ansatz2 Komplexität und Performanz, Vergleich, Einfärbung der Gleichung Ansatz1, Legende maMo Ansatz2

FRAGEN[GG]: mathematische Modelle rausnehmen?

Abstract: Durch die äußerst effiziente Nutzung von Hardware mittels Virtualisierung steigt die Nachfrage nach virtualisierten Infrastrukturen enorm. Serviceprovider müssen die Hardwarebasis für ihre Dienste nicht mehr selbst unterhalten und geben ihre dahingehende Verantwortung an Infrastrukturanbieter weiter. Um die Sicherheit dieser Strukturen nicht zu vernachlässigen, arbeiten viele Forscher in diesem Bereich und versuchen effiziente Algorithmen mit integrierter Beachtung der Sicherheitsaspekte zu finden. Diese Arbeit soll einen Überblick und eine Klassifizierung der Gefahren, die solche Konstrukte betreffen, sowie eine Analyse zweier unterschiedlicher Ansätze zur Vermeidung möglichst vieler Risiken zum Zeitpunkt der Planung übermitteln.

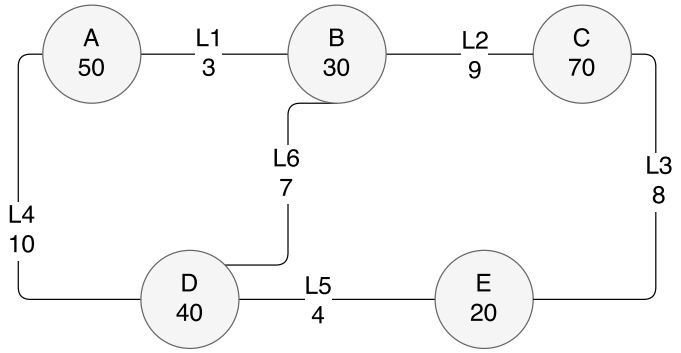
Inhaltsverzeichnis

1 Einleitung

2 Das Newtork Embedding Problem

Um den Anforderungen der heutigen Gefahren zu genügen, ist es unumgänglich sämtliche Grundsätze der IT-Sicherheit möglichst früh in die Planung der gewünschten Infrastruktur miteinzubeziehen. Nicht nur, weil das nachträgliche Schließen von Sicherheitslücken und Hinzufügen von Sicherheitskomponenten in finanzieller und zeitlicher Hinsicht zehnmal so teuer ist wie die initiale Beachtung dieser Aspekte, sondern weil die vollständige Sicherheit eines nachgerüsteten Systems kaum gewährleistet werden kann [Col11].„Security-by-Design“ ist einer der wichtigsten Begriffe bei der Planung öffentlich zugänglicher Netzwerkstrukturen. Dementsprechend groß ist die Nachfrage nach VNE-Algorithmen, die bereits beim Prozess des Mappings möglichst viele Sicherheitsaspekte beachten und abdecken. Die vorausgehende Klassifizierung der bekannten Gefahren in „VNE-relevant“ und „nicht-VNE-relevant“ bildet die Grundlage für unsere weiteren Untersuchungen. Um zuvor noch einen genaueren Einblick in die Problematik von VNE zu gewähren, beginnen wir zunächst mit der Erläuterung des VNE-Problems sowie einem Überblick über 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, erhält man die Basismenge

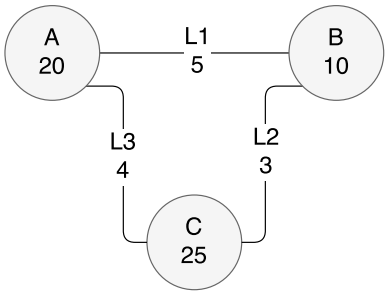
$$G^S = \{N^S, L^S\}$$

(N steht für die Menge der physischen Knoten, L für 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 übersetzt werden.

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

(N steht für die Menge der virtuellen Knoten, L für 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 üblich, nicht nur eine einzelne VNRs auf ein physisches System abzubilden, sondern gleich mehrere auf einmal. Ebenso wäre es theoretisch möglich, dass ein Infrastruktur-Provider gleich mehrere getrennte unabhängige physische Strukturen betreibt, und somit mittels VNE das beste System für die Abbildung der Menge von VNRs eroiren möchte. Dies könnte beispielsweise der Fall sein, wenn ein Serviceprovider ein VNR mit der Bedingung „alle Elemente mögen sich am selben Standort befinden, egal an welchem“ in Auftrag gibt, und der Infrastruktur-Provider über mehrere Hardware-Standorte verfügt.
Die Attributmenge bei den grundlegenden VNE-Algorithmen beschränkt sich zumeist auf Rechnerleistung und Bandbreite. Lokalität, GPU-Leistung und RAM-Menge wären einige weitere mögliche Standard-Attribute. Ein großes Problem bei der Berechnung des optimalen Mappings findet sich bei der Berechnungszeit. Die endliche Beschränkung der Knoten- sowie Link-Resourcen und die „on-line nature“ von VNRs stellen zusätzliche Hindernisse dar. Da selbst Lösungen für einfache Anfragen (geringe Anzahl von Knoten und Links), welche wenige Attribute beinhalten, exponentiellen Rechenaufwand benötigen, steigt der Aufwand sowohl mit der Anzahl der abzubildenden Knoten und Links, als auch mit steigender Attributanzahl dementsprechend. Teilweise gilt das Problem als rechnerisch unlösbar, grundsätzlich aber befinden wir uns im Komplexitätsbereich der NP-Vollständigkeit [MRR13]. Da die Erhöhung der Attributmenge - wie bereits genannt - grundsätzlich nicht positiv zur Laufzeit der existierenden Algorithmen beiträgt, werden

wir den Komplexitätsbereich beim Hinzufügen von Sicherheitsanforderungen nicht verlassen. Dennoch gibt es Möglichkeiten, die den Rechenaufwand reduzieren können. Im Folgenden widmen wir uns allerdings zuerst einer Übersicht über die Gefahren, welche bei VNE eine Rolle spielen.

3 Sicherheitsaspekte bei virtuellen Netzwerk-Infrastrukturen

3.1 Herkömmliche Gefahren

3.2 Spezielle Gefahren bei virtualisierten Umgebungen

3.3 VNE-Relevante Gefahren

4 Vermeidung von Gefahren via Secure VNE (SVNE)

Da wir nun einen Überblick zu VNE sowie den bestehenden Gefahren gegeben haben, widmen wir uns nun zwei verschiedenen SVNE-Lösungsansätzen.

4.1 Ansatz 1

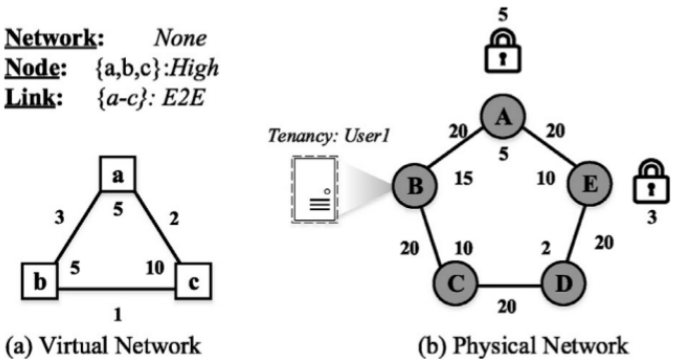
Hierbei beschäftigen wir uns mit dem Lösungsansatz aus [WCC16]. Das Hauptaugenmerk bezüglich der Sicherheitsaspekte legen Wang et al auf Traffic-Verschlüsselung und die Separierung von VMs unterschiedlicher Trust-Levels. Mittels„Security plan design“werden die drei folgenden strukturellen Aspekte betrachtet 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 für sich. Hiermit soll die Wahrscheinlichkeit für DOS-Angriffe oder ähnliche vermindert werden, da„Mehr-Parteien-Netzwerke“die Hauptschwachstellen für solche Angriffe darstellen [Liu10]. Auch Sniffing durch kompromittierte Hosts im selben Netz wird durch diese Maßnahme verhindert. Während ”high“Resourcenteilung somit komplett verweigert, lässt der Level„medium“zumindest ein gemeinsam genutztes Netz für VNRs vom selben Eigentümer 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“lässt VNR-Knoten vom selben Eigentümer auf ein und

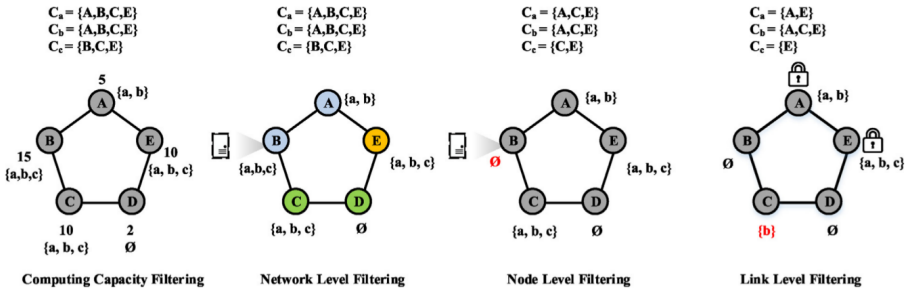
demselben physischen Knoten zu. Durch diesen Plan sollen Angriffe von VM zu VM über gemeinsam genutzte Resourcen unterbunden werden. Zusätzlich wird das Risiko eines Angriffs vom physischen Host verringert, da der Angriffsvektor„VM zu physischem Host“eliminiert wird.

- The Link plan:
  End-to-End(E2E), Point-to-Point(P2P) und„none“sind die hier wählbaren Levels. Während E2E nur an den Endpunkten Verschlüsselungskapazitäten zu Verfügung stellen muss, benötigt P2P diese Kapazitäten an allen Hops des abgebildeten Links. Die heutzutage sehr gängigen man-in-the-middle Angriffe sollen dadurch wesentlich erschwert werden.

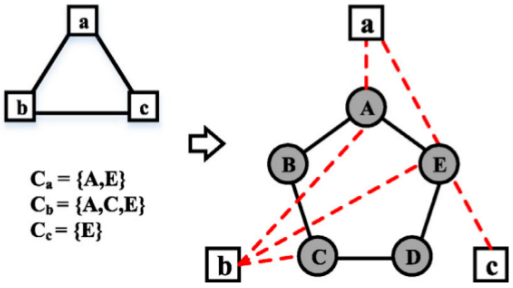
VNRs werden nun, wie in Kapitel 2 bereits gezeigt, in ungerichtete Graphen mit Standardanforderungentransformiert. Zusätzlich werden hier auch noch Sicherheitsanforderungen integriert.



Nun wird ein vier-stufiges Pre-Processing durchgeführt, um die Berechnung der optimalen Abbildung der VNRs zu vereinfachen. Hierbei werden als ersten die Standardanforderungen der virtuellen Knoten mit den zu Verfügung stehenden Kapazitäten der physischen Knoten verglichen. Sollten physische Knoten bestimmte Anforderungen nicht erfüllen, 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 genügt 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 Verschlüsselungsanforderungen der virtuellen Links nicht genüge tun.



Mit den aus dem Pre-Processing gewonnen Information bildet man nun einen Hilsgraphen, welcher die übrig gebliebenen Möglichkeiten 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 Fälle unterschieden.„Path-splitting”, im weiteren SVNE-PS und„no-path-splitting”, im weitern SVNE-NPS, sind zusätzliche Sicherheitsvorgaben, welche im Vorfeld definiert werden müssen, um die Wahl des Algorithmus zu ermöglichen. Sollte SVNE-PS gewählt werden, beziehen sich die Gleichungen nur auf die Erfüllung der geforderten Attribute. Sollte SVNE-NPS gewählt werden, werden die für 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.

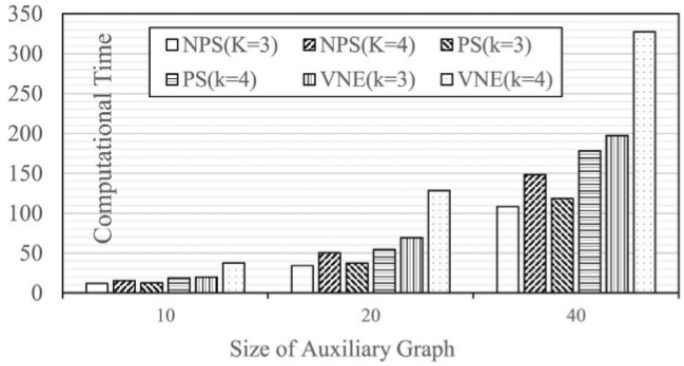
Fehlt: Kurze Beschreibung der Punkte 1-13

Die Berechnungen wurden mittels induktiver logischer Programmierung unter Verwendung von CPLEX[hi] und durch k-shortest-path anhand aller abzubildenden Elemente limitiert. Für die Testumgebungen wurden zufällig erzeugte VNRs mit 2 bis 10 Knoten und halbsovielen Links erzeugt. Auch die Rechenkapazität wie Bandbreite wurden durch Zufallsgeneratoren mit Werten im Bereich von 1 bis 10 gewählt und verteilt. Die Anzahl der VNRs beträgt 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 zufällig erzeugt und beinhalteten 10 bis 50 Knoten, sowie halbsoviele Links. Die Rechen- sowie Bandbreiten-Kapazitäten wurden gleichmäßig verteilt und enthielten Werte zwischen 1 und 50. Die Ver-

schlüsselungskapazitäten der Knoten wurde über die gesamte Testreihe ebenfalls gleichmäßig verteilt.

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



Man sieht hier einen deutlichen Zeitvorsprung von PS und NPS gegenüber dem Standard-Algorithmus. Da hier jedoch auch der Standard-Algorithmus vom Pre-Processing profitiert, wäre ein Vergleich, welcher die Dauer des Pre-Processings mit aufnimmt und den Standard-Algorithmus ohne Pre-Processing arbeiten ließe, anschaulicher und aussagekräftiger. Dennoch ist ein Punkt durchaus überraschend und bemerkenswert: Der Standard-Algorithmus benötigt deutlich länger, obwohl er keine Sicherheitsaspekte mitbeurteilt, im Gegenzug zu seinen Kontrahenten.

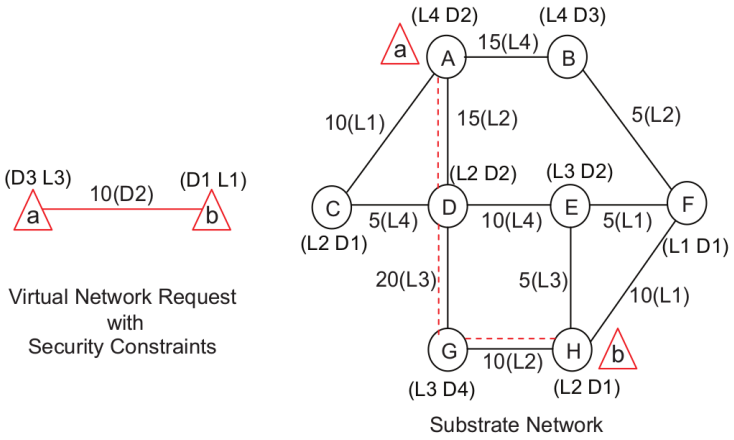
In welchem Komplexitätsbereich sich das Pre-Processing befindet wird hier leider nicht erwähnt.

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 hierfür verwendet, was allerdings nicht zwingend so vorgesehen ist. Stattdessen wäre eine Verwendung komplexerer Sicherheitsvektoren möglich, welche bei weitem detailliertere Sicherheitsmerkmale beschreiben könnten. Des weiteren verfügt jeder physische wie auch virtuelle Knoten über zwei Sicherheitswerte: Anforderungslevel und (eigenes) Sicherheitslevel. Der Anforderungslevel definiert das Minimum-Level des Gegenübers, der Sicherheitslevel definiert die eigenen gewünschten Sicherheitsmerkmale. Virtuelle Links verfügen über Anforde-

rungslevels, physische Links nur über Sicherheitslevels. Die vorausgesetzten Basisanforderungen, welchen alle VNRs unterliegen, beschränken sich auf 4 Regeln, welche mittels der zuvor genannten Merkmale umgesetzt werden:

- Ein physischer Knoten sollte einen Sicherheitslevel garantieren, der höher ist als die Anforderungen der darauf abzubildenden virtuellen Knoten.
- Der Sicherheitslevel des virtuellen Knotens sollte höher sein, als das Anforderungslevel des physichen Knotens.
- Alle virtuellen Knoten, welche auf den selben physischen Knoten abgebildet werden, sollten über einen ausreichenden Sicherheitslevel verfügen.
- 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 durchgeführte Abbildung berücksichtigt nicht nur die Sicherheitsanforderungen aller Parteien unter Berücksichtigung der Basisanforderungen, sondern achtet zusätzlich noch auf Kostenminimierung. Insgesamt existieren drei Abbildungsmöglichkeiten, die gewählte ist jedoch die günstigste, unter dem Aspekt keine Sicherheitsresourcen zu verschwenden. Dieser VNR hätte auch auf EDGH abgebildet werden können, wobei der Link ED über einen Sicherheitslevel verfügt, welcher höher als nötig ist. Würde dies nicht beachtet werden, könnte es zu unnötigen Engpässen bei der Behandlung von VNRs mit höheren Anforderungslevels kommen. Um ein solches Verhalten zu erreichen, wurden zusätzlich 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

Der zugehörige Abbildungsalgorithmus bietet nun zwei Optionen: uSAV und cSAV. uSAV ist für den laufenden Betrieb, also die Abbildung bestehender VNRs sowie der Abbildung neu hinzugekommener VNRs zuständig und bildet diese zeitsparend ab. Das heißt die optimale Abbildung wird hier zwar selten erreicht, dafür wird unter Betrachtung der Kosten-/Nutzen-Berechung ein günstiges Ergebnis erzielt. cSAV arbeitet, im Gegenzug zu uSAV, auf optimale Abbildungen hin. Um dies zu erreichen trennt cSAV Knoten- und Link-Abbildungen strikt, was zu enormen Verschlechterungen bei "no-path-splitting-Anforderungen führt. Beide Algorithmen arbeiten mit den selben Heuristiken und Sub-Algorithmen: Node Mapping und Link Mapping

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.

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

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`.

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`.

4.3 Vergleich

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 führt zu Leistungseinbußen. [CDRS07]

Literatur

[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.

[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.