Telematics
Computer Systems

Freie Universität Berlin

Computer Systems and Telematics

Bachelor-Arbeit

Vergleichende Analyse von Abwehrmethoden gegen Interest Flooding Attacks in Named Data Networking

Samir Al-Sheikh

Matr. 4457630

Betreuer: Dr. Emmanuel Baccelli
Dipl.-Inf. Matthias Wählisch

Institut für Informatik, Freie Universität Berlin, Deutschland

9. Januar 2014

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, sind als solche gekennzeichnet. Die Zeichnungen oder Abbildungen sind von mir selbst erstellt worden oder mit entsprechenden Quellennachweisen versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner Prüfungsbehörde eingereicht worden.

Berlin, den 9. Januar 2014

(Samir Al-Sheikh)

Abstract

Die vorliegende Arbeit analysiert vergleichend verschiedene Abwehrmethoden gegen Interest-Flooding-Attacks innerhalb von Named Data Networking (NDN). Ausgehend von der Architektur und Funktionsweise von NDN wird die Vorgehensweise einer Interest-Flooding-Attack dargestellt. Aktuelle Lösungsansätze werden detailliert diskutiert. Inkonsistenzen in bestehenden Protokollbeschreibungen werden aufgedeckt und bereinigt. Die Abwehrmethoden werden in einem Simulator implementiert und evaluiert, wobei die Implementierungen gegenüber bestehenden Untersuchungen verifiziert werden. Durch die Verwendung eines einheitlichen Simulators wird die Vergleichbarkeit der erzielten Ergebnisse sichergestellt. Die in dieser Arbeit gewonnenen Ergebnisse zeigen, dass eine Abwehr, die auf einem einzigen Kriterium basiert, Defizite bei der Identifizierung des Angreifers aufweist. Um eine Lösung für diese Schwachstelle zu bieten, wurde eine Abwehrmethode, basierend auf zwei Kriterien entwickelt. Hierdurch lässt sich eine eindeutige Erkennung des Angreifers ermöglichen und somit die Angriffsabwehr effizienter umsetzen. Die vorliegende Arbeit schließt mit einer Zusammenfassung der Ergebnisse und einer Aussicht auf anstehende Herausforderungen bei der Weiterentwicklung der Abwehrmethoden ab.

This work carefully compares current defense methods against Interest-Flooding-Attacks within Named Data Networking (NDN). Based on the architecture and operation of NDN we discuss the problem of Interest-Flooding-Attack. We explain recently proposed countermeasures in detail, reveal inconstencies in current protocol descriptions, and revise them. Implementing the approaches consistently in a single simulator, we analyse and discuss performance measures. The simulations are verified with existing studies. In this work, we find that a defense protocol based on a single criterion has shortcomings in identifying the attacker. To offer a solution to this vulnerability, we develop an approach which is applies two criteria. This allows for unique identification of the attacker. Finally, this increases efficiency of countermeasures againt Interest Flooding Attack. This paper concludes with a summary of the findings and upcoming challenges in the development of defense methods.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

KAPITEL 1

Einleitung

Seit seinen Anfängen vor mehr als 35 Jahren hat sich das Internet zur erfolgreichsten Kommunikationsinfrastruktur der Welt entwickelt. Es verbindet inzwischen mehr als 100 Millionen Menschen miteinander und ist aus vielen Bereichen des Lebens nicht mehr wegzudenken, wobei die Anwendungsgebiete kaum noch Grenzen haben.

Die ursprüngliche Kommunikationsform des Internets basiert auf dem Ende-zu-Ende-Prinzip [5] der Netzwerkschicht. Der globale Austausch von Daten erfolgt zwischen den IP-Schnittstellen der involvierten Teilnehmer. Höherstehende Anwendungen können dies abstrahieren, benötigen jedoch separate Dienste, die eine Zuordnung zwischen Daten und Endgerät vornehmen. Die dabei verwendeten Routing-Wege im Internet werden unabhängig von einem Kommunikationsbedarf im Voraus erstellt.

Die Nutzung des Internets hat sich über die Jahrzehnte stark verändert, z.B. durch multimediale Datenverteilung, Kommunikation über soziale Netzwerke, dass sich das Nutzungsmodell zunehmend von einer Ende-zu-Ende-Kommunikation hin zu einer Ende-zu-Daten-Kommunikation entwickelt hat. Internetnutzer sind an einem schnellen Datenzugriff interessiert, unabhängig davon, wo die Daten ursprünglich publiziert wurden. Webinhalte beispielsweise liegen nicht mehr ausschließlich auf einem Webserver bereit, sondern werden global über Content Delivery Networks [6] verteilt, um geringere Latenzen zum Endnutzer zu erzielen.

Die Anforderungen neuer Anwendungen und Dienste, aber auch Sicherheitsaspekte haben vor einigen Jahren das Forschungsfeld des *Future Internet* motiviert [7]. Einer dieser Ansätze ist das Named Data Networking (NDN) [1], [8], [9]. NDN unterstützt auf der Netzwerkschicht nicht nur das Routing auf Daten-Namen, sondern auch In-Network-Caching. Die darunterliegende Routing-Technologie basiert auf einem Publish/Subscribe-Ansatz. Dadurch werden Kommunikationswege zwischen Teilnehmern – im Gegensatz zum heutigen Internet – erst bei Bedarf etabliert.

Der in NDN verfolgte Ansatz bietet zwei Vorteile: (a) Daten werden automatisch im Netzwerk repliziert, wobei das Netzwerk die Lokalisierung des Speicherorts übernimmt, (b) Endsysteme sind nicht mehr adressierbar, wodurch klassische Distributed-Denial-of-Service-Angriffe (DDoS) [10] unwirksam werden. Gleichzeitig öffnet jedoch das Publish/Subscribe-Routing [11] die Control Plane der Netzwerkinfrastruktur für den Endnutzer [12, 13] und

erhöht damit die Verwundbarkeit eines zukünftigen Internet Backbones. Der Interest-Flooding-Angriff versucht die Router-Ressourcen auszuschöpfen, um zeitweise eine Unterbrechung der Datenverteilung zu erzielen [15]. Aktuelle Abwehrmethoden [4], [16] versuchen den NDN-Router davor zu schützen, wobei diese auf jeweils anderen Angriffserkennungen und Angreifer-Rückverfolgungen basieren. Obwohl die Anzahl der Anti-Interest-Flooding-Ansätze stetig wächst, existiert bisher keine vergleichende Analyse der Lösungsansätze. In der vorliegenden Arbeit wird dieses Defizit aufgegriffen und eine simulative, vergleichende Analyse für alle bisher existierenden Abwehrmethoden gegen einen Interest-Flooding-Angriff vorgenommen. Der weitere Verlauf der Arbeit gliedert sich wie folgt:

Das nachfolgende Kapitel beschreibt die Funktionsweise und die Architektur von Named Data Networking und erläutert die Angriffsart Interest Flooding Attack. In Kapitel 3 werden einzelne Abwehrmethoden und deren Vorgehensweise gegen eine Interest Flooding Attack an Beispielen vorgestellt. Kapitel 4 beschreibt den Entwurf des Simulators [17] und die Implementierungen der Abwehrmethoden. Die jeweiligen Simulationen werden im darauf folgenden Kapitel verifiziert. In Kapitel 6 wird eine vergleichende Evaluierung vorgenommen, deren Auswertung in Kapitel 7 erfolgt. In einem abschließenden Kapitel werden die Ergebnisse der vorliegenden Arbeit zusammengefasst und ein Ausblick auf mögliche weitere Entwicklungen gegeben.

KAPITEL 2

Hintergrund

2.1 Named Data Networking

Information-Centric Networking (ICN) [18], [19], [20] ist das Ergebnis aktueller Netzwerk-Forschungen und stellt einen Ansatz dar, das jetzige Internet mit Hilfe eines Ende-zu-Daten-Konzepts zu realisieren. Dabei werden Informationsobjekte und deren Eigenschaften genutzt, um eine bessere Datenverteilung zu ermöglichen [20].

Named Data Networking (NDN) [8] ist ein Ansatz für die Umsetzung einer künftigen Internetarchitektur, der auf ICN basiert. Es agiert als ein empfängergesteuertes Protokoll, da eine Interaktion nur auf Anfrage des Empfängers stattfindet. Es basiert außerdem auf der Architektur von Content Centric Networking (CCN) [21]. Statt einer Adressierung der Endpunkte einer Verbindung, werden ausschließlich Daten mit Hilfe von Namen adressiert. Diese Namen werden ähnlich wie IP-Adressen hierarchisch geordnet und bestehen aus für den Menschen lesbaren Symbolen, von Trennzeichen „/“ unterteilt, z.B. „/ieee/videos/1“. Dabei gibt es einen Präfixteil „/ieee/videos“ und einen Segmentteil „/1“. Aus einer Ende-zu-Ende-Kommunikation, wie sie im jetzigen Internet stattfindet, bei der der Daten-Interessent den Daten-Ersteller nach den Daten fragt, wird eine Ende-zu-Daten-Kommunikation bei der der Daten-Interessent (Consumer) das Netzwerk um den Daten-Ersteller (Producer) nach dem Datennamen suchen lässt. Das Netzwerk sucht dabei die naheliegendste Kopie der Daten und leitet diese dem Daten-Interessenten. Falls keine Daten im Netzwerk existieren, wird der Producer gefragt. Es entsteht eine Unabhängigkeit zwischen Daten und Ersteller-Speicherort, wodurch mehrere Paradigmen des Internets vereinfacht bzw. erleichtert werden.

In NDN gibt es zwei verschiedene Typen von Paketen: Interest und Data. Wenn ein Consumer Daten anfragt, verschickt dieser ein Interest in Richtung des Producers. Diese Pakete werden als 1-zu-1-Beziehung behandelt, d.h. pro Interest gibt es nur ein Data-Paket. Außerdem werden Data-Pakete nur als Antwort für Interests geschickt und nehmen durch die Gegebenheiten der Router denselben Weg wie das dazugehörige Interest (Reverse Path Forwarding). Dieses Interest enthält Informationen über gewünschte Datennamen (Content Name), einige Auswahlmöglichkeiten (Selector) und den gewünschten Bereich (Nonce), aus dem die Daten stammen sollen sowie einer Lebenszeit ähnlich der momentan verwendeten Time-To-Live.

Falls das Interest bis zum Producer mehrere Knoten (Router) überwindet, wird „unterwegs“ bereits geprüft, ob die gewünschten Daten verfügbar sind (Network-Caching). Falls das der Fall ist, werden diese bereits dort als Data-Paket verschickt. Ein Data-Paket enthält - analog zum Interest - Informationen über den Datennamen, eine Signatur, wodurch der Producer eindeutig erkennbar ist, signierte Informationen und die eigentlichen Daten.

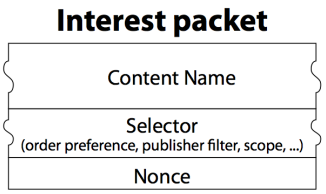

Abbildung 2.1: Spezifikation eines Interest-Pakets [1]

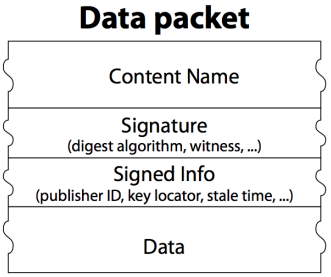

Abbildung 2.2: Spezifikation eines Data-Pakets [1]

Jeder Router besitzt mehrere Interfaces, über die Pakete ausgetauscht werden können. Bei Empfang eines Interests gibt es eine vorgeschriebene Nutzungsabfolge der Datenstrukturen. Dabei werden die einzelnen Datenstrukturen, die nachfolgend näher beschrieben werden, mit Hilfe der Longest-Prefix-Suche untersucht.

Knoten (Router) zwischen Consumer und Producer können als Weiterleitung und Zwischenspeicher genutzt werden. Damit dies funktioniert gibt es innerhalb jedes Routers drei verschiedene Datenstrukturen: Content Store (CS), ein Speicher der Network-Caching ermöglicht. Pending Interest Table (PIT) zur Speicherung auf Daten wartender Interests. Forwarding Information Base (FIB), die eine Auflistung aller Datenquellen darstellt.

Der Content Store (CS) fungiert als Zwischenspeicher für bereits empfangene Data-Pakete und verhält sich nach dem LRU-Prinzip (Least-Recently-Used). Data-Pakete, die oft gefordert werden, werden am längsten gespeichert. Dadurch „rücken“ bei vielfachen Anfragen, die Daten scheinbar zum Benutzer hin, da jeder weitere Router auf dem Weg ebenfalls die Daten speichert. Anfangs wird geprüft, ob es zu einem eingehenden Interest ein passendes Data-Paket im Content Store gibt. Dies trifft nur zu, wenn beide Content Names identisch sind. Falls dies der Fall ist, wird auf dem empfangenen Interface das entsprechende Data-Paket geschickt. Wenn innerhalb des Content Store kein entsprechendes Data-Paket existiert, wird die Pending Interest Table daraufhin untersucht.

In der Pending Interest Table (PIT) werden alle Interests vermerkt, die ein dazugehöriges Data-Paket erwarten. Es wird der angefragte Content Name des Interests und das entsprechende Interface in der Tabelle erfasst. In der PIT wird analog zum Content Store ebenfalls nach dem Content Name gesucht. Falls ein Eintrag existiert und das anfragende Interface dasselbe ist, wird das eingehende Interest verworfen. Ein Eintrag wird solange behalten, bis ein passendes Data-Paket angekommen, die Lebenszeit des Interests oder der Timer der PIT abgelaufen ist.

Anschließend wird die letzte Datenstruktur untersucht, die Forwarding Information Base (FIB). Diese enthält alle dem Router bekannten Namenspräfixe und deren ausgehende Interfaces. Wenn ein Eintrag gefunden wurde, wird das dazugehörige Interest in die PIT

eingetragen. Nun wird das Interest an das passende Interface gegeben und somit weitergeleitet. Falls kein Eintrag existiert, gibt es keine passenden Daten zum Interest oder keinen verfügbaren Pfad zur Datenquelle. Das Interest wird dann verworfen.

Named Data Networking verfügt über eine Strategie-Schicht, die die Auswahl des Interfaces in der FIB verbessern soll. Somit kann jeder Router in bestimmten Intervallen prüfen, welches Interface für bestimmte Daten in der FIB am schnellsten antwortet, um möglichst geringe Verzögerungen zu erreichen. Ein weiterer Vorteil der NDN-Architektur ist die mit inbegriffene Sicherheit (Security by Design), die im jetzigen Internet nicht existiert. So muss nicht mehr die gesamte Verbindung verschlüsselt werden, sondern nur noch die eigentlichen Daten, die der Consumer als Antwort auf ein Interest erhält. Diese werden mit Hilfe einer Signatur vom Producer an den Content-Namen gebunden und gewährleisten so eine universelle Authentizität des Pakets. Dabei ist irrelevant, wann und wo das Paket empfangen wird. Durch die beschriebene Vorgehensweise werden bereits existierende Angriffsmöglichkeiten wie z.B. der DDoS-Angriff, unschädlich gemacht, da es keine direkte Verbindung mehr zwischen Angreifer und Ziel gibt.

Dennoch könnten die zuvor genannten Vorteile von NDN für neuartige Angriffsmöglichkeiten missbraucht werden, wie zum Beispiel: Interest Flooding Attack [13], [16], [17], [22].

2.2 Interest Flooding Attack

In den letzten Jahren kamen Angriffe wie Denial-of-Service (DoS) und Distributed-Denial-of-Service (DDoS) vermehrt vor und richteten zunehmende Schäden an. Die heutige DDoS-Methode wäre durch die Gegebenheiten von NDN jedoch nicht mehr möglich, da keine Ende-zu-Ende-Paradigmen existieren. Jedoch ermöglichen genau diese Gegebenheiten eine neue Variante des DDoS-Angriffs, die Interest Flooding Attack [14]. Diese nutzt das Zwischenspeichern aller wartenden Interests innerhalb der PIT im Router aus. Ein oder mehrere Angreifer generieren eine Vielzahl verschiedener Interests und füllen so die PIT. Somit ist es dem Router nicht mehr möglich, Interests von legitimen Benutzern ordnungsgemäß weiterzuleiten, wodurch einzelne Router, Teile des Netzwerks um den Producer oder der Producer selbst nicht mehr funktionieren können.

Man unterscheidet zwischen drei verschiedenen Angriffstypen: [12]

- existierende Interests (statisch),
- dynamisch generierte Interests (dynamisch),
- dynamisch nicht existierende Interests.

Beim statischen Versenden von existierenden Interests werden sehr viele Interests für dasselbe Datum verschickt – ähnlich dem heutigen DDoS. Diese Methode ist sehr ineffizient, da in NDN oft angefragte Daten im Content Store gespeichert werden und dem Angreifer „entgegenkommen“. Da die angefragten Daten bereits existieren, werden die entsprechenden PIT-Einträge bei Dateneingang gelöscht. Dadurch entfernt sich der Angriff mit fortlaufender Dauer immer weiter von dem Datenersteller.

Bei einem dynamischen Angriff werden sehr viele dynamisch generierte Interests versendet, die verschiedene Daten anfragen. Dadurch entfällt der Vorteil des Zwischenspeicherns im

Content Store und das Angriffsziel bleibt während des gesamten Vorgangs der Producer. Analog zum ersten Typ werden die PIT-Einträge beim Eintreffen von Daten gelöscht. Da für den Angreifer das dynamische Generieren von existierenden Daten jedoch aufwendiger als das statische Versenden, reduziert dieser Angriffstyp nur einige Ressourcen, wird jedoch nicht in der Lage sein, alle Ressourcen für sich zu beanspruchen.

Bei dem dritten Angriffstyp werden dynamisch nicht existierende Interests generiert. Ähnlich wie zuvor, kann der Content Store den Router nicht vor Schaden bewahren. Jedoch ist bei dieser Methode das Generieren der Interests nicht aufwändig, da einfach nicht existierte Daten angefragt werden und der Sequenz-Teil zufällig gewählt werden kann. Wenn der Präfix-Teil existiert, der Sequenz-Teil beim Producer aber auf keinerlei Daten verweist, so werden alle Interests direkt zum Producer weitergeleitet. Auch werden diese in der PIT nicht auf übliche Weise (durch Datenerhalt) gelöscht. Sie existieren solange, wie es die Lifetime-Information im Interest vorsieht bzw. bis der Timer des Routers abläuft. Dadurch ist es dem Angreifer möglich, die Interests solang wie möglich in der PIT zu halten. Der Angreifer erreicht sein gewünschtes Ziel - die enorme Auslastung der PIT des Routers.

Im Nachfolgenden wird sich stets auf letzten Angriffstyp bezogen, da er das größte Angriffspotenzial birgt [16].

KAPITEL 3

Abwehrmethoden

Da sich NDN noch in der Entwicklungsphase befindet, werden zeitgleich Methoden entwickelt, die eine Interest Flooding Attack abwehren könnten. Diese Strategien sollen das Weiterleiten von Interests in jedem Router regulieren (Forwarding Strategy). Eine naive Lösung gegen Angriffe könnte das Begrenzen der PIT auf einen stetig berechneten Wert sein. Wenn dieser Wert erreicht wird, werden alle weiteren Interests verworfen und nicht weitergeleitet. Dabei entsteht jedoch das Problem, dass ein möglicher Angreifer genau diese PIT-Größe mit seinen Interests ausfüllen könnte und alle weiteren Interests von legitimen Consumern verworfen werden, was den Effekt des Angriffs nur verstärken würde. Im folgenden Abschnitt werden verschiedenen Ansätze, die versuchen, solche Nebeneffekte zu vermeiden, dargestellt und an Beispielen erläutert.

In diesen Beispielen werden folgende zwei Topologien benutzt:

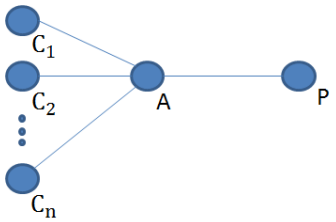

Abbildung 3.1: Topologie für Abwehrmethoden-Beispiele mit einem Router

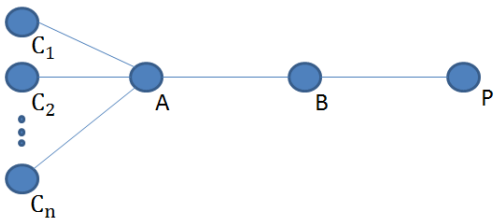

Abbildung 3.2: Topologie für Abwehrmethoden-Beispiele mit zwei Routern

3.1 Interest Flooding Attack and Countermeasures in Named Data Networking

In dem Artikel „Interest Flooding Attack and Countermeasures in Named Data Networking“ [2] werden drei verschiedene Arten der Abwehr gegen Interest Flooding Attacks vorgestellt:

1. Token bucket with per interface fairness,
2. Satisfaction-based Interest acceptance,
3. Satisfaction-based pushback.

Token bucket with per interface fairness

Bei dieser Abwehrmethode wird neben der PIT-Größe die Reihenfolge der weitergeleiteten Interests betrachtet und reguliert. Hierfür wird die bestehende PIT erweitert. Für jedes eingehende Interface wird eine Queue erstellt, in der die Interests eingetragen werden. Die Weiterleitung der Interests (Tokens) erfolgt über die Auswahl des zu sendenden Interfaces mithilfe des Round-Robin-Prinzips, indem in einer bestimmten Reihenfolge jedes Interface eine vorgegebene Anzahl an Interests verschicken darf.

Damit die Information, ob ein Interest in einer Queue wartet oder bereits weitergeleitet wurde, verfügbar ist, wird die bestehende PIT um die Spalte „Status“ erweitert und bidirektionale Pointer gesetzt. Somit können die zu verbrauchenden Ressourcen und die Zeit, die ein Interest in einer Queue verbleiben darf, vom Router festgelegt werden.

Der Zeitstrahl in Abbildung 3.3 stellt zu aufeinander folgenden Zeitpunkten den Zustand in der PIT eines Routers dar. Dort befindet sich neben den herkömmlichen Spalten „Prefix“, „in“ und „out“ eine weitere Spalte „Status“. Zum Zeitpunkt t_1 werden auf einem Router sechs Interests empfangen.

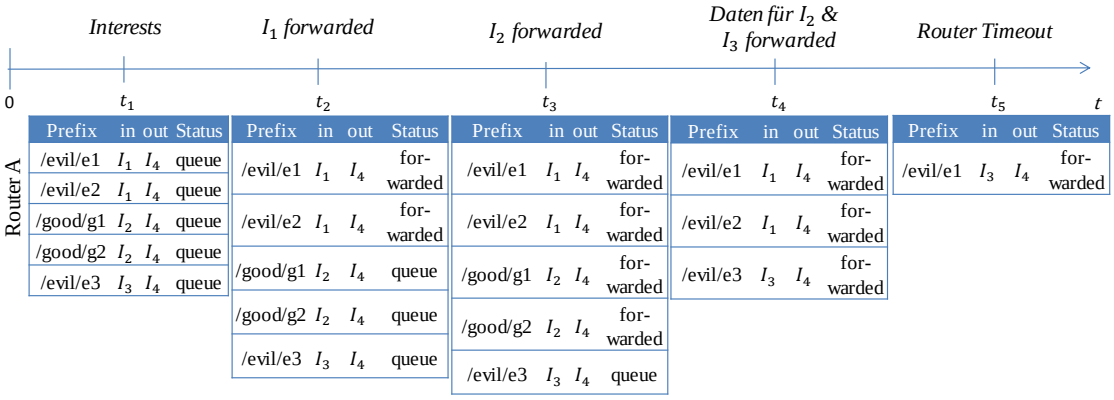

Abbildung 3.3: Zeitstrahl für „Token bucket with per interface fairness” PIT-Berechnung für jedes Interface I_i eines Routers

Diese werden gemäß des herkömmlichen Routerverhaltens in die PIT eingetragen. Außerdem befinden sich alle Einträge zu diesem Zeitpunkt in der jeweiligen Queue. Zum Zeitpunkt t_2 werden zwei Interests des Interfaces I_1 weitergeleitet. Im nächsten Schritt werden ebenfalls zwei Interests des Interface I_2 weitergeleitet. Zu diesen Interests folgen bei t_3 die jeweiligen Daten, wodurch die PIT-Einträge entfernt werden. Außerdem werden nun Interests des Interfaces I_3 weitergeleitet. Zum Zeitpunkt t_5 ist der Router-Timeout für die Interests von I_1 abgelaufen, was zur Löschung der PIT-Einträge führt.

Dieses Abwehrverfahren ermöglicht eine relative Fairness unter den eingehenden Interfaces an jedem Router, da alle Interests gleich behandelt werden. Das Problem ist jedoch, dass die Absolutmenge an Interests von legitimen Consumern und Angreifern in der Regel nicht „fair“ ist, sondern die Anzahl der angreifenden Interests sehr viel höher liegt [2].

Satisfaction-based Interest acceptance

Damit eine Strategie effektiv ist, muss diese zwischen angreifende und legitimen Interests unterscheiden [2]. Um dies zu ermöglichen, benutzt diese Methode ein anderes Auswahlkriterium für die Differenzierung der Interests. Statt alle Interfaces und somit alle Consumer mittels Round-Robin gleich zu behandeln, wird nun eine Wahrscheinlichkeit für das Weiterleiten eines Interests berechnet. Die Weiterleitungswahrscheinlichkeit eines Interfaces ist als das Verhältnis von eingehenden Interests und erfolgreich beantworteten Interests. Laut Autoren ergibt sich bei einem vermeintlich angreifenden Interface ein geringerer Wahrscheinlichkeitswert, da keine beantworteten Interests existieren. Bei legitimen Consumern wird hingegen ein großer Wert ermittelt. Dementsprechend werden Interests legitimer Consumer mit einer höheren Wahrscheinlichkeit weitergeleitet. Jeder Router fällt so anhand der lokal vorliegenden Wahrscheinlichkeiten die Entscheidung, ob ein Interest weitergeleitet wird oder nicht.

Abbildung 3.4 beschreibt die Vorgehensweise der Abwehrmethode anhand der PIT-Einträge innerhalb eines Zeitintervalls. Zum Zeitpunkt t_1 erhält der Router A mehrere Interests auf verschiedenen Interfaces und trägt diese in die PIT ein. Die Wahrscheinlichkeit für eine Weiterleitung wird für jedes Interface auf 100 % initialisiert. Dementsprechend werden alle Interests entsprechend der FIB an den nächst gelegenen Router B gesendet. Anschließend werden die Daten für den Präfix „/good“ empfangen. Sobald der PIT-Timeout erfolgt ist, können die Wahrscheinlichkeiten neu berechnet werden. Da auf dem Router A alle Interests des Interfaces I_2 Daten erbracht haben, erhält dieses Interface 100 %, die anderen 0 %. Analog erhält das Interface I_1 des Routers B eine Wahrscheinlichkeit von 50 %. Zum Zeitpunkt t_4 erhält der Router A nur noch Interests des Interfaces I_2, welche ebenfalls an Router B weitergeleitet werden. Dieser nimmt von diesen drei Interests nur 50 % an. Es bleiben die zuvor berechneten Wahrscheinlichkeitswerte erhalten, bis Daten empfangen werden oder der Router-Timeout erfolgt.

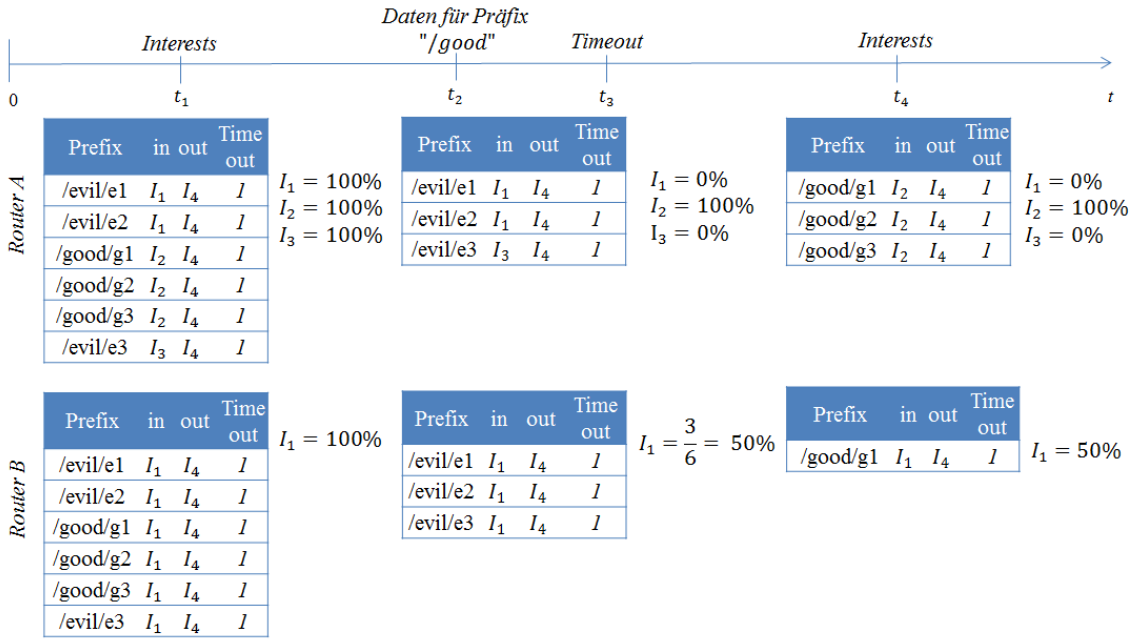

Abbildung 3.4: Zeitstrahl für „Satisfaction-based Interest acceptance"
PIT-Berechnung für jedes Interface I_i eines Routers

Die Struktur dieser Methode ermöglicht eine Unterscheidung von legitimen und angreifenden Consumern. Das Problem dieser Methode ist die zu Grunde gelegte Berechnung der gesamten Weiterleitungswahrscheinlichkeit eines Interests auf dem Weg bis zum Producer. Da die gesamte Wahrscheinlichkeit das Produkt jeder lokalen Wahrscheinlichkeit ist, nimmt die Wahrscheinlichkeit, dass ein Interest den Producer erreicht, mit jedem weiteren Router ab [2].

Satisfaction-based pushback

Um das Problem der abnehmenden Wahrscheinlichkeit eines Interests über den gesamten Transportweg zu lösen, werden bei der Satisfaction-based-pushback-Abwehrmethode die Entscheidungen der Router an benachbarte Router weitergegeben. Statt Wahrscheinlichkeiten werden absolute Limit-Werte für jedes eingehende Interface festgelegt, die die Anzahl der erlaubten Interests explizit bestimmen.

Der Producer setzt ein Anfangslimit für alle eingehenden Interfaces fest und sendet dieses an seinen dahinter liegenden Router. Durch eingehende Interests entstehen nun - analog zu Satisfaction-based Interest acceptance - in diesem Router Wahrscheinlichkeitswerte für jedes Interface. Um die Weiterleitungswahrscheinlichkeit zu aktualisieren, wird diese nun mit dem davor erhaltenen Limit des Producers multipliziert. Der aktualisierte Wert wird an die benachbarten Router geleitet bis alle Router ihre Werte aktualisiert haben. Die Abbildung 3.5 zeigt einen Ausschnitt aus einer möglichen Verbindung, bei der die Pushback-Methode angewendet wird.

Auch bei dieser Abwehrmethode erhalten die Router A und B zum Zeitpunkt t_1 mehrere

Interests und initialisieren die Wahrscheinlichkeiten mit 100 %. Der Weiterleitungsfaktor (Limit-Wert) des Producers wird auf 10 gesetzt. Bei der Berechnung dieser Werte multipliziert der Router, der erste nach dem Producer, die Wahrscheinlichkeiten mit diesem Weiterleitungsfaktor.

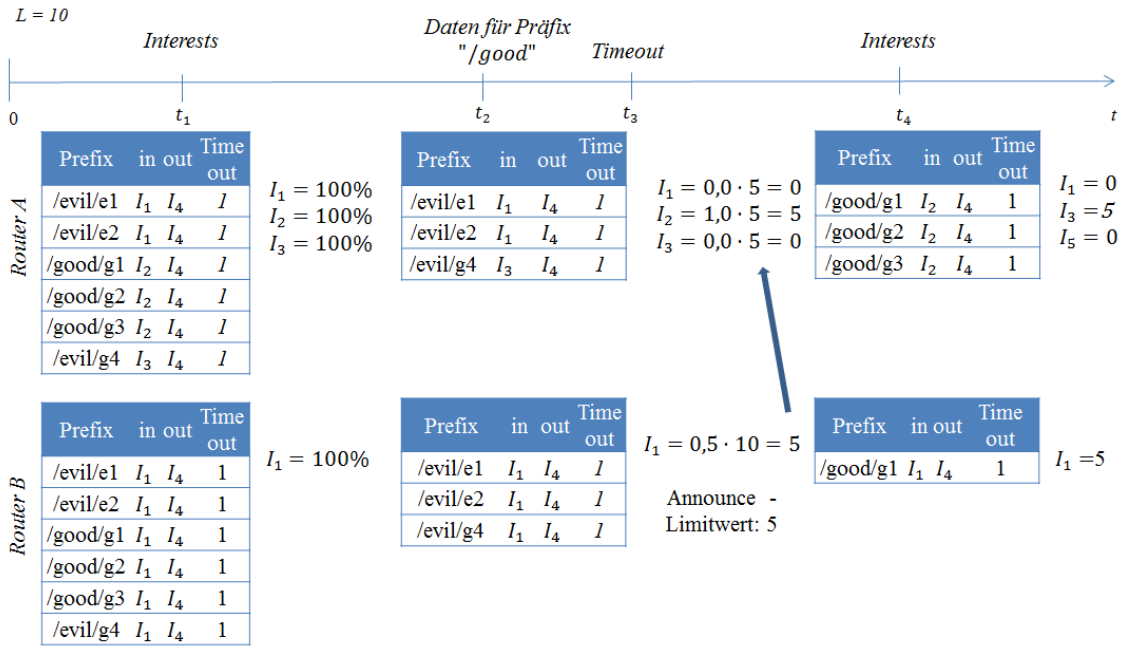

Abbildung 3.5: Zeitstrahl für „Satisfaction-based pushback"
PIT-Berechnung für jedes Interface I_i eines Routers

Zum Zeitpunkt t_2 erhalten die Router Daten für das Präfix „/good", was zur Löschung der dazugehörigen PIT-Einträge führt. Zum Zeitpunkt t_3 erfolgt nun der Router-Timeout, wodurch die restlichen PIT-Einträge entfernt werden und die Wahrscheinlichkeiten für jedes Interface feststehen. Für das Interface I_1 des Routers B entsteht ein Wert von 5, da 50 % der Interests datenbringend waren. Somit dürfen auf diesem Interface maximal fünf Interests zeitgleich weitergeleitet werden. Dieser Limit-Wert wird nun an den darauf folgenden Router, Router A, geschickt. Dieser multipliziert diesen Wert ebenfalls mit allen Wahrscheinlichkeiten der Interfaces, wodurch das Interface I_2 alle fünf Interests für sich beanspruchen darf. Interests der Interfaces I_1 und I_3 werden hingehend nicht weitergeleitet. Nach diesen Werten werden die Interfaces nun bei weiteren Interests limitiert, wie zum Zeitpunkt t_4 ersichtlich. Die Limit-Werte werden neu berechnet, nachdem Daten empfangen wurden oder der Router-Timeout erfolgt ist.

3.2 DoS and DDoS in Named Data Networking

In dem Artikel „DoS and DDoS in Named Data Networking" [16] wird eine Abwehrmethode gegen eine Interest Flooding Attack vorgestellt, die auf der „Pushback"-Vorgehensweise basiert. Da diese dort namentlich nicht benannt wird, wird diese im Folgenden „DDoS-

Abwehrmethode“ genannt.

Bei dieser Methode wird ein bestehender Angriff erkannt, in dem der PIT-Verbrauch jedes Präfixes eines Interfaces überprüft wird. Jeder weiterzuleitende Präfix pro Interface in der PIT gilt als Bedrohung, wenn die Anzahl der empfangenen Interests bei diesem Präfix das Limit überschreitet.

Als Reaktion auf einen Angriff, wird der Präfix limitiert, indem ein bestimmter Prozentsatz der eingehenden Interests verworfen wird. Außerdem wird der benachbarte Router, der sich hinter dem angreifenden Interface befindet, alarmiert. Dieser limitiert lokal ebenfalls den angreifenden Präfix und leitet den Alarm weiter in Richtung Angreifer. Dieses Verfahren setzt sich rekursiv fort. Letztlich blockiert der letzte Router des NDN-Netzwerks ebenfalls diesen Präfix, so dass keine weiteren Interests dieses Präfixs ins NDN-Netzwerk gelassen werden [16].

Die Abbildung 3.6 zeigt eine Beispielverbindung, bei der diese Limitierung benutzt wird. Zum Zeitpunkt t_1 empfängt der Router A eine Vielzahl von Interests auf unterschiedlichen Interfaces.

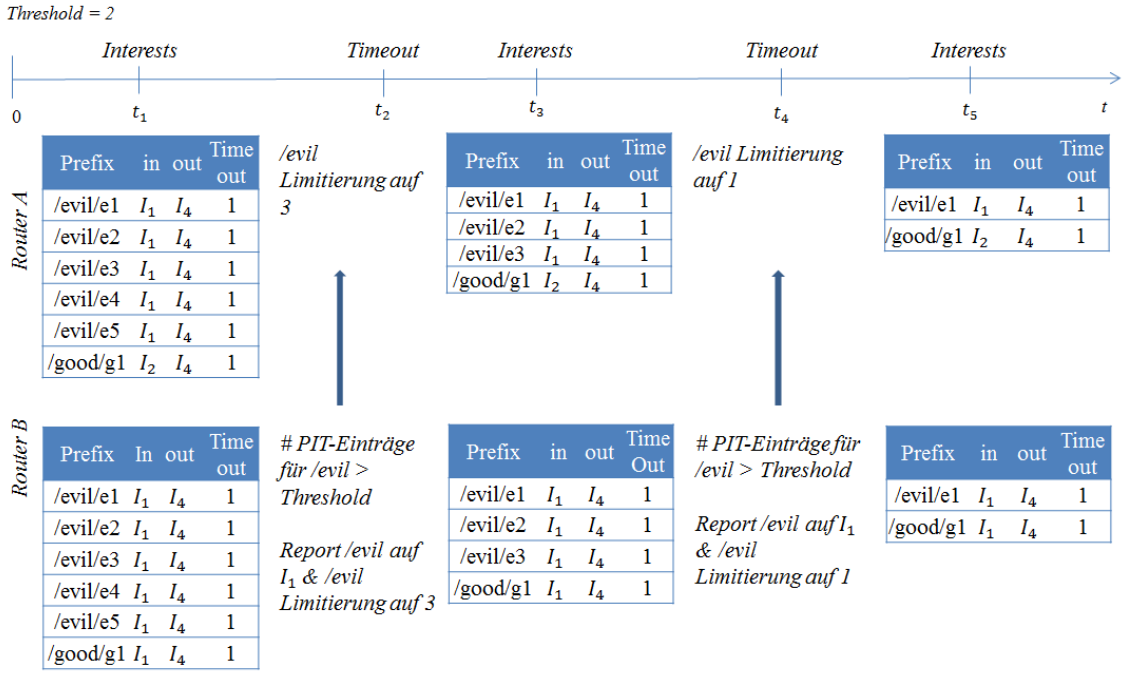

Abbildung 3.6: Zeitstrahl für „Dos and DDoS in Named Data Networking”
PIT-Berechnung für jedes Interface I_i eines Routers

Diese werden wie üblich an Router B weitergeleitet. Dieser erkennt einen Angriff, da die Anzahl der PIT-Einträge für den Präfix „/evil“ auf dem Interface I_1 größer als der hier festgelegte Limit-Wert ist (Threshold = 2). Dadurch wird der Präfix - wie bereits erklärt - limitiert (50 %) und ein Report an das angreifende Interface gesendet. Router A erhält nun dieses Paket und limitiert den Präfix „/evil“ ebenfalls auf 50%. Nachdem nun der Router-Timeout erfolgt ist, werden weitere Interests empfangen. Da die Anzahl der PIT-

Einträge immer noch größer als der Limit-Wert ist, wird erneut limitiert und ein Report versandt. Router A erhält dieses Paket und limitiert ebenfalls. Zum Zeitpunkt t_4 erfolgt erneut ein Router-Timeout. Bei t_5 empfängt der Router weitere Interests, wobei die Anzahl der PIT-Einträge nun unterhalb des Limit-Wertes liegt, wodurch keine weitere Limitierung erforderlich ist.

3.3 Mitigate DDoS Attacks in NDN by Interest Traceback

In dem Artikel „Mitigate DDoS Attacks in NDN by Interest Traceback“ [3] wird die Abwehr von Interest Flooding Attacks durch „Interest Traceback“ beschrieben. Dabei wird ein momentaner Angriff erkannt, wenn die Größe oder der Anstieg der Größe der PIT im Router einen bestimmten Wert erreicht. Wenn dies der Fall ist, wird die Interest-Traceback-Methode aktiviert. Dabei werden alle Interests, für die seit langer Zeit keine Daten angekommen sind, erfasst. Anschließend werden für diese Interests gefälschte Daten (Spoofed-Data-Pakete) simuliert und wie normale Data-Pakete an die Angreifer zurückgeschickt. Diese Spoofed-Data-Pakete enthalten den gleichen Inhalt wie die jeweiligen Interests und werden von weiteren Routern ebenfalls wie normale Daten weitergeleitet. Sobald das Datenpaket den letzten NDN-Router erreicht hat, mit dem der Angreifer über ein Interface verbunden ist, wird dieses Interface vom Router auf eine bestimmte Rate limitiert. Somit gelangen je nach Limitierungsrate nur noch wenige angreifende Interests in das NDN-Netzwerk [3].

In Abbildung 3.7 ist ein mögliches Szenario dargestellt. Zum Zeitpunkt t_1 erhält Router A mehrere Interests auf zwei verschiedenen Interfaces I_1 und I_2. Diese werden dann an Router B weitergeleitet, um mögliche Daten vom Producer zu erhalten.

Router B erkennt nun einen möglichen Angriff, da die Anzahl der PIT-Einträge größer als der hier festgelegte Limit-Wert (PIT-Threshold = 5) ist. Außerdem ist die Rate an eingehenden Interests pro Zeitsegment größer als der Wert des Limits (PIT-Rate = 5). Als Folge wird der Interest-Traceback-Mechanismus aktiviert. Noch vor dem Router-Timeout werden Spoofed-Data-Pakete an die Interfaces geschickt, die zu den bisher datenlosen PIT-Einträgen gehören. Router A, der nun der letzte Router des NDN-Netzwerkes ist, erhält diese Pakete und drosselt die dazugehörigen Interfaces auf einen fest definierten Wert, zum Beispiel 2. Nachdem der Router-Timeout erfolgt, werden weitere Interests empfangen, wobei pro Interface jeweils nur zwei Interests toleriert werden. Die Limit-Werte werden nicht mehr überschritten und lösen keine weiteren Spoofed-Data-Pakete aus.

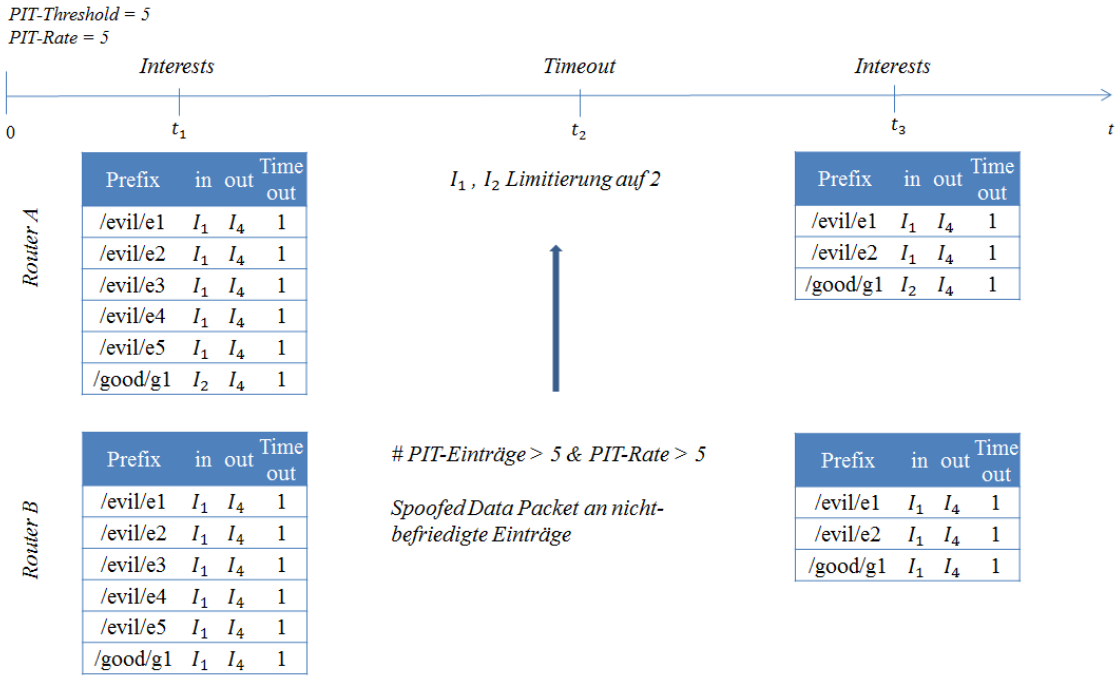

Abbildung 3.7: Zeitstrahl für „Mitigate DDoS Attacks in NDN by Interest Traceback" PIT-Berechnung für jedes Interface I_i eines Routers

3.4 Poseidon - Mitigating Interest Flooding DDos Attacks in NDN

In dem Artikel „Poseidon - Mitigating Interest Flooding DDos Attacks in NDN" [4] werden drei Methoden erläutert, um eine Interest Flooding Attack abzuwehren. Dabei werden proaktive und reaktive Abwehrmethoden unterschieden. Voraushandelnde Methoden verhindern, dass es zu einem bedrohenden Angriff im NDN-Netzwerk kommt. Ein Beispiel für eine solche Methode wird mit dem Resource-Allocation-Verfahren vorgestellt. Bei den rückwirkenden Methoden wird auf einen bereits erfolgten oder momentan noch aktiven Angriff reagiert und versucht, ihn abzuwenden. Beispiele dafür sind das Poseidon-Local- und das Poseidon-Distributed-Verfahren.

Die genannten Verfahren werden im folgenden Abschnitt vorgestellt.

Resource Allocation

Bei dieser Methode wird eine Maximalgröße der PIT (mpu) festgelegt. Diese wird stetig neu berechnet: $mpu = rate \cdot timeout$ [4]. Dabei ist $rate$ die Summe aller eingehenden Interests in einem Zeitintervall für alle Interfaces und $timeout$ der Timeout-Wert der Interests in der PIT. Wenn die PIT den festgelegten Maximalwert erreicht, werden alle folgenden Interests verworfen. Die Abbildung 3.8 zeigt ein mögliches Beispiel für das Router-Verhalten mit Resource Allocation [4].

Die Maximalgröße mpu wird mit einem Wert von 6 Interests initialisiert. Zum Zeitpunkt t_1 erhält der Router A diverse Interests auf verschiedenen Interfaces. Nun wird die mpu neu berechnet. Da die Anzahl der momentanen PIT-Einträge größer ist als dieser Wert, wird die PIT-Größe entsprechend limitiert. Nachdem das Router-Timeout alle keine Daten er-

bringenden Interests entfernt hat und alle Daten bringenden PIT-Einträge gelöscht wurden, erhält der Router A erneut Interests. Dabei wird nur eine Interestsanzahl gestattet, die der *mpu* entspricht.

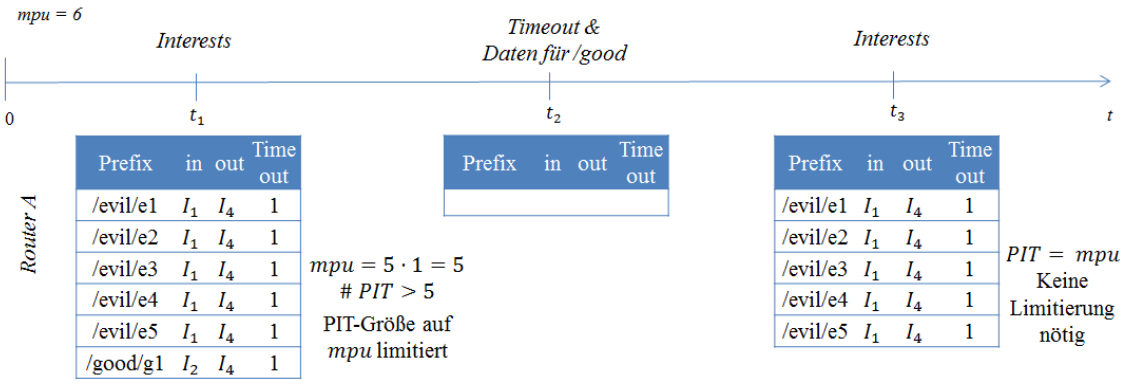

Abbildung 3.8: Zeitstrahl für Resource Allocation"
PIT-Berechnung für jedes Interface I_i eines Routers

Poseidon-Verfahren

Das Poseidon-Verfahren ist eine rückwirkende Methode, wobei der Angriff zunächst erkannt und anschließend abgewehrt werden muss. Die Methode umfasst also zwei verschiedene Phasen: die Erkennungsphase (Detection Phase) und die Reaktionsphase (Reaction Phase). Bei letztgenannter unterscheidet man in Poseidon Local, wobei ausschließlich der angegriffene Router agiert, und Poseidon Distributed, bei dem mehrere Router als verteiltes System innerhalb des NDN-Netzwerks dem Angriff entgegenwirken.

Poseidon Local

Um einen Angriff im NDN-Netzwerk zu erkennen, werden innerhalb jedes Routers kontinuierlich zwei Parameter berechnet: $\omega(r_i^j, t_k), \rho(r_i^j, t_k)$.

$\omega(r_i^j, t_k)$ ermittelt das Verhältnis von eingehenden Interests zu eingehenden passenden Daten an einem Interface j im Router r_i im Zeitintervall t_k. [4]

$$\omega(r_i^j, t_k) = \frac{\text{(Anzahl von Interests von } r_i^j \text{ in } t_k)}{\text{(Anzahl von Datenpaketen von } r_i^j \text{ in } t_k)}$$

$\rho(r_i^j, t_k)$ berechnet die Größe der PIT-Einträge für das Interface j im Router r_i im Zeitintervall t_k. Somit entsteht folgende Berechnung für den Wert: [4]

$$\rho(r_i^j, t_k) = \text{Größe der PIT in Bytes von } r_i^j \text{ in } t_k$$

Falls diese beiden Parameter im selben Zeitintervall, meist jede 60 ms, ihre jeweiligen Limits $\Omega(r_i^j)$ und $P(r_i^j)$ überschreiten, wird im Router ein Angriff verzeichnet und die Reaktionsphase eingeleitet. Somit ist $\omega(r_i^j, t_k)$ ein Indikator, inwiefern dieser in der Lage ist, die ankommenden Interests mit passenden Daten zu versorgen. Dieser Wert kann jedoch bei kürzeren Interest-Stößen – was bei legitimen Consumern der Fall sein kann – schnell steigen. Wenn dies als Angriff interpretiert würde, könnte dies die Netzwerk-Performance verschlechtern. Um diesen Effekt zu vermeiden und die Effizienz der Messung zu erhöhen, wird zeitgleich $\rho(r_i^j, t_k)$ im Router gemessen. Dieser Wert gibt an, wie viel Platz ein bestimmtes Interface innerhalb der PIT einnimmt. Bei den genannten Interest-Stößen würde dieser Wert nicht sein Limit erreichen und so eine falsche Interpretation verhindern.

Sobald ein Angriff verzeichnet wird, wird das Interface gedrosselt, dessen Werte die Limits überschritten haben, indem weitere Interests verworfen werden. Dadurch kann ein Angriff von einem spezifischen Interface erkannt und gebannt werden, ohne legitime Consumer zu benachteiligen. Fallen die Werte nach einem Angriff wieder unter die entsprechenden Limits, wird die Drosselung aufgehoben [4].

In Abbildung 3.9 wird ein mögliches Szenario für einen Interest-Flooding-Angriff, der mithilfe von Poseidon Local abgewehrt werden könnte, dargestellt.

Zum Zeitpunkt t_1 empfängt Router A mehrere Interests, die auf zwei verschiedenen Interfaces ankommen. Daraufhin werden die zwei Parameter $\omega(r_i^j, t_k)$ und $\rho(r_i^j, t_k)$ für die Interfaces I_1 und I_2 berechnet.

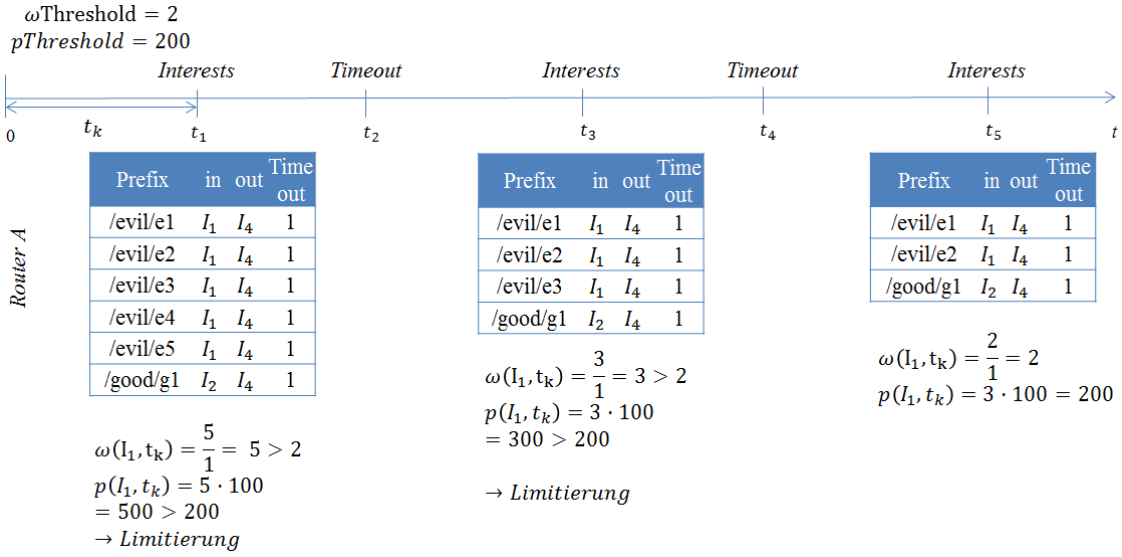

Abbildung 3.9: Zeitstrahl für „ Poseidon Local"
PIT-Berechnung für jedes Interface I_i eines Routers

Da der errechnete Wert $\omega(I_1, t_k)$ größer als der fest definierte Limit-Wert (ωThreshold $= 2$) und der $\rho(r_i^j, t_k)$-Wert größer als pThreshold ist, wird eine Limitierung des betroffenen Interface auf einen bestimmten Prozentsatz (50 %) eingeleitet. Nachdem der Router-Timeout erfolgt ist, erhält der Router zum Zeitpunkt t_3 weitere Interests, wobei die Anzahl auf die Hälfte beschränkt ist. Dennoch werden erneut die Parameterwerte berechnet. Diese sind

größer als die jeweiligen Limit-Werte, wodurch eine erneute Limitierung eingeleitet wird. Nach wiederholtem Router-Timeout überschreiten die danach empfangenen Interests nicht die vorgegebenen Limitwerte, wodurch keine Limitierung mehr nötig ist.

Da es sich bei dem diesem Ansatz um eine lokale Abwehrmethode handelt, kann im Fall eines Angriffes nicht genau gesagt werden, ob der potenzielle Angreifer nun vollständig aus dem NDN-Netzwerk ausgeschlossen wurde. Um dieses Problem zu lösen, gibt es eine Erweiterung des lokalen Ansatzes, den verteilten Ansatz – das Poseidon-Distributed-Verfahren.

Poseidon Distributed

Diese Abwehrmethode ermöglicht es, den Angriff bis zum Angreifer zurückzudrängen, um diesen zeitweise vollständig aus dem Netzwerk zu entfernen. Um dies zu realisieren, wird bei einem erkannten Angriff das potenziell angreifende Interface gedrosselt. Außerdem wird ein Alarm-Paket über das angreifende Interface in Richtung Angreifer verschickt. Dieses Datenpaket wird zum Nachrichtenaustausch genutzt, da dies vom benachbarten Router empfangen werden kann, unabhängig vom momentanen Zustand der dortigen PIT. Außerdem sind diese Datenpakete im Gegensatz zu Interests signiert, wodurch Falschmeldungen vermieden werden können. Das Datenpaket kann nur in die Richtung vom Producer zum Consumer verschickt werden (einseitige Kommunikation). Es enthält den Präfix „/pushback/alerts/“ und wird vom Router nicht wie ein reguläres Datenpaket behandelt. Zusätzlich enthält es den vom Angreifer gebrauchten Präfix der Interests, die Erstellungszeit des Pakets und die gedrosselte Datenrate des angreifendes Interfaces. Bei Eingang eines solchen Pakets werden die Limits $\Omega(r_i^j)$ und $P(r_i^j)$ verringert, bis der Angriff ebenfalls erkannt wird. Daraufhin wird das angreifende Interface auf diesem Router gedrosselt, so dass nur ein bestimmter Prozentsatz an Interests akzeptiert wird [4]. Danach wird das Alarmpaket weiter verschickt. Somit erreicht das Paket schließlich den letzten zu dem NDN-Netzwerk gehörenden Router, wo ebenfalls die Interests-Rate heruntergesetzt wird. Falls über eine bestimmte Zeitspanne keine Angriffe mehr über dieses Interface wahrgenommen werden, werden die originalen Werte wiederhergestellt. Dadurch sind Router, die kein direktes Ziel des Angriffs darstellen, jedoch angreifende Interests weiterleiten, in der Lage, Angriffe zu erkennen und abzuwehren [4].

In Abbildung 3.10 wird – ähnlich wie bei dem lokalen Ansatz – ein mögliches Szenario für einen Interest-Flooding-Angriff dargestellt, wobei der verteilte Ansatz angewendet wird. Im Router A, der den letzten Router des NDN-Netzwerks darstellt, werden die Limit-Werte ωThreshold auf 6 und pThreshold auf 600 und im Router B auf ωThreshold = 3 und pThreshold = 300 initialisiert. Zum Zeitpunkt t_1 empfängt Router A Interests auf zwei Interfaces, die an Router B weitergeleitet werden. Dort werden die Parameter analog zum lokalen Ansatz berechnet. Da diese die Limitwerte des Routers überschreiten, wird die Reaktionsphase eingeleitet. Das entsprechende Interface wird auf den definierten Prozentsatz von 50% limitiert. Des weiteren wird ein Alert-Paket an das angreifende Interface verschickt, welches Router A kurz darauf erhält. Dieser verringert nun solange seine Limit-Werte, bis er den entsprechenden Angriff wahrgenommen hat. Nachdem der Router-Timeout erfolgt ist bzw. die PIT-Einträge datenbringender Interests gelöscht wurden, gehen erneut eine Vielzahl an Interests bei beiden Routern ein. Durch die lokale Limitierung in der vorigen Iteration stellt Router B nun keine Gefahr durch das angreifende Interface mehr fest. Da Router A seine Limit-Werte aus dem vorigen Schritt verringert hat, wird der Angriff nun

erkannt und ebenfalls limitiert. In der nächsten Iteration, nach erfolgtem Router-Timeout, erhalten beide Router erneut Interests, die jedoch unter den Limit-Werten liegen, wodurch keine weitere Limitierung mehr nötig ist.

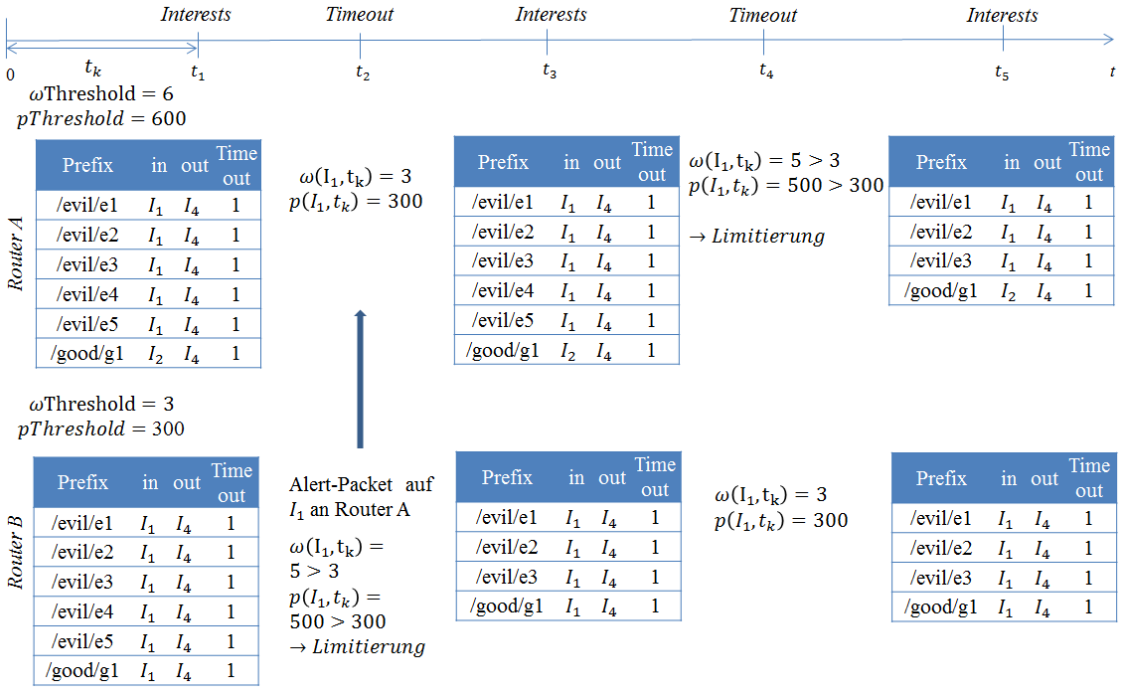

Abbildung 3.10: Zeitstrahl für „Poseidon Distributed"
PIT-Berechnung für jedes Interface I_i eines Routers

3.5 Detecting and Mitigating Interest Flooding Attacks in Content-Centric Network

In dem Artikel „Detecting and mitigating interest flooding attacks in content-centric network" [23] wird eine weitere Methode zur Abwehr von Interest-Flooding-Angriffen vorgestellt. Da die Angriffserkennung ausschließlich auf Limits (Thresholds) basiert, wird sie als „threshold-based detecting and mitigating" (TDM) bezeichnet.

Dabei basiert die Abwehrmethode auf folgender Annahme: Da angreifende Interests (MI) im Gegensatz zu legitimen Interests (OI) keine Daten zurückbringen und somit der entsprechende PIT-Eintrag im Router abläuft, wird die Anzahl der abgelaufenen PIT-Einträge als Indikator für einen Angriff betrachtet.

Die herkömmliche FIB wird um vier Parameter M, C_I, F_S, und Capacity erweitert. Der Modus M beschreibt die Art des eingehenden Interests (MI, OI). Der Zähler C_I misst die Anzahl fortlaufender Interests mit gleicher Art. F_S ermittelt, wie oft C_I den dazugehörigen Threshold C_{th} überschreitet. Der *Capacity*-Wert gibt die für die Weiterleitung zugelassene Anzahl an Interests pro Präfix an. Dadurch wird jeder Eintrag der FIB, somit jeder Präfix,

durch genau diese vier Werte charakterisiert. Die Interests werden in zwei Modi: OI und MI unterteilt. Der Modus des momentan eingehenden Interests für diesen Präfix wird in M gespeichert. Somit nimmt M den Wert OI an, sobald ein legitimes Interests und analog MI, wenn ein angreifendes Interest empfangen wird. C_I ermittelt nun die Anzahl an Interests zu diesem Präfix, die hintereinander eintreffen und den gleichen Modus besitzen. Falls ein Interest mit einem anderen Modus eintrifft, wird C_I auf 1 zurückgesetzt. C_I kann damit als Indikator für mögliche angreifende Interests genutzt werden. Ein festgesetztes Limit C_{th} soll helfen, einen Angriff zu erkennen. Der Wert F_S spiegelt den Status des Angriffes wider, ob dieser noch besteht und behandelt werden muss oder bereits abgewehrt wurde. Wenn der Wert von C_I das Limit C_{th} überschreitet und der Modus des dafür entscheidenden Interests OI war, wird der Wert von F_S um 1 inkrementiert. Wenn der Wert von C_I das Limit C_{th} überschreitet und der Modus des dafür entscheidenden Interests allerdings MI war, wird der Wert auf 0 gesetzt. Falls der Wert F_S unterhalb des Limits F_{th} liegt, ist der Angriff auf dem Router noch aktiv und der Capacity-adjusting-Mechanismus wird eingeleitet. Falls F_S jedoch ein entsprechendes Limit F_{th} erreicht, ist der Angriff vorbei, da mehr OI registriert sind als MI, so dass der Capacity-adjusting-Mechanismus gestoppt wird. Der Parameter Capacity legt fest, wie viele Interests eines Präfixes von dem Router weitergeleitet werden dürfen. Der EnableCapacityAdjusting-Wert (ECA) zeigt, ob diese Limitierung momentan aktiv ist oder nicht [23].

In der Abbildung 3.11 wird ein mögliches Szenario von eingehenden Interests dargestellt, die Teil eines Interest-Flooding-Angriffs sind.

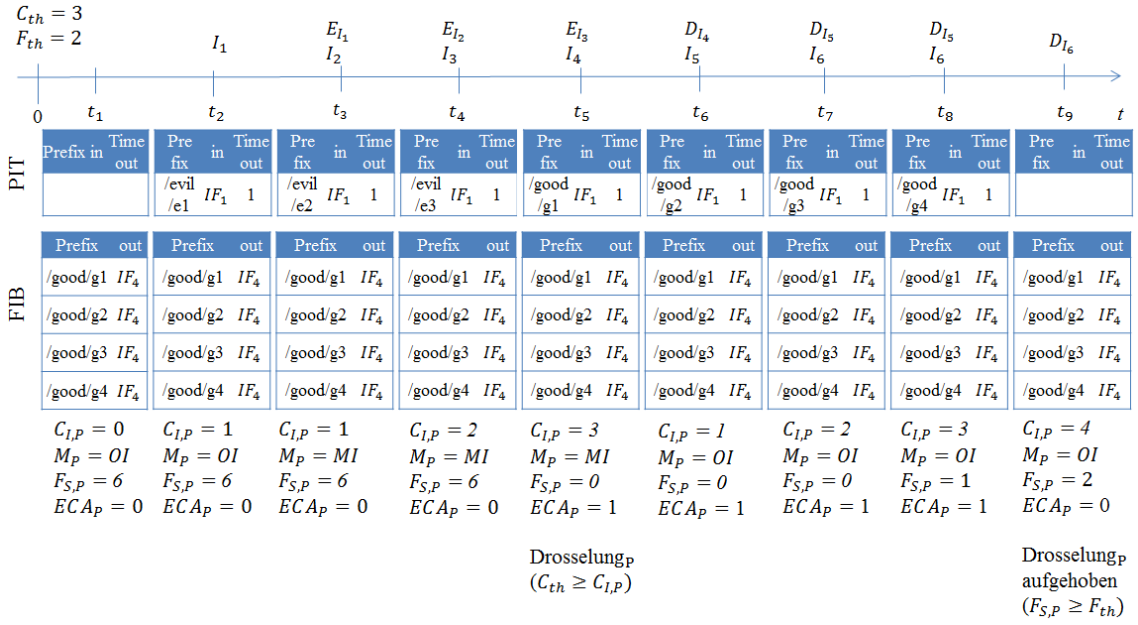

Abbildung 3.11: Zeitstrahl für die Status-Berechnung in „Detecting and mitigating interest flooding attacks in content-centric network" bei eingehenden Interests I_i für jeden Präfix P in der FIB_R des Routers R

Die Limit-Werte werden auf $C_{th} = 3$ und $F_{th} = 2$ initialisiert. Außerdem wird $F_{S,P}$ auf 6 gestellt, wodurch anfangs kein Angriff wahrgenommen wird. Zum Zeitpunkt t_1 werden keine Interests empfangen, wodurch die initialisierten Werte bestehen bleiben und die PIT ungefüllt ist. Im nächsten Schritt empfängt der Router ein Interest, wobei dieser davon standardmäßig ausgeht, dass es sich dabei um ein legitimes Interest (OI) handelt. Dadurch wird der Zähler $C_{I,}P$ um 1 inkrementiert. Der Modus M_P nimmt den Status des Interests OI an. Der Capacity-adjusting-Mechanismus ist somit deaktiviert ($ECA_P = 0$). Zum Zeitpunkt t_3 läuft das Interest aufgrund von fehlenden Daten aus, wodurch der PIT-Eintrag gelöscht wird. Dadurch wird davon ausgegangen, dass es sich bei diesem Interest um ein angreifendes Interest (MI) handelt. Der Zähler wird auf den Wert 1 zurückgesetzt und der Modus $M_P = MI$ gesetzt. Außerdem erhält der Router ein weiteres Interest. Dieses läuft im Zeitpunkt t_4 ebenfalls ab, wodurch $C_{I,P}$ inkrementiert wird und der Modus bei MI bleibt. der Capacity-adjusting-Mechanismus bleibt weiterhin deaktiviert. Ein weiteres Interest wird empfangen, das analog zu seinen Vorgängern im Zeitpunkt t_5 abläuft. Dadurch wird $C_{I,P}$ erneut inkrementiert und erreicht den Limit-Wert 3. Der Modus bleibt unverändert. Der Capacity-adjusting-Mechanismus wird aktiviert und somit das Interface auf eine bestimmte Anzahl an Interests gedrosselt. Außerdem wird ein Interest empfangen, was zum Zeitpunkt t_6 Daten erbringt, wodurch der Zähler auf den Wert 1 zurückgestellt wird. Der Modus geht in den Zustand OI. In den nachfolgenden Zeitpunkten empfängt der Router bis t_8 weitere datenbringende Interests auf diesem Interface, wobei der Capacity-adjusting-Mechanismus stets aktiviert bleibt. Zum Zeitpunkt t_8 erreicht der Zähler $C_{I,P}$ erneut den Limit-Wert, wobei M_P jedoch auf OI gesetzt ist. Dadurch wird $F_{S,P}$ um 1 inkrementiert.Im letzten Zeitpunkt erreicht dann der Zähler $F_{S,P}$ seinen Limit-Wert von 2 wodurch die Drosselung aufgehoben bzw. der Capacity-adjusting-Mechanismus deaktiviert wird.

3.6 Erweiterung: Interface and Prefix-based Countermeasure

Die Abwehrmethode „Interface and Prefix-based Countermeasure“ (IPC) ist ein Verfahren zur Erkennung, Rückverfolgung und Abwehr eines Interest-Flooding-Angriffs. Um dies zu ermöglichen, werden zunächst die PIT-Einträge aller Interfaces in bestimmten Zeitintervallen auf Größe geprüft. Überschreitet dabei eines dieser Interfaces eine fest definierte Obergrenze, wird dies als Angriff gedeutet und der Präfix mit den meisten PIT-Einträgen für dieses Interface ermittelt. Anschließend wird ein Alarm-Paket mit dem potenziell angreifenden Präfix in Form eines Datenpakets an das entsprechende Interface gesendet und der Präfix auf diesem Interface blockiert. Anschließend wird das Alarm-Paket weitergeleitet, zunächst an den nachfolgenden Router. Dieser blockiert ebenfalls diesen Präfix auf dem Interface, das die PIT-Obergrenze überschreitet, und schickt dann das Paket weiter in Richtung Angreifer, so dass letztendlich der Angriff aus dem NDN-Netzwerk gedrängt wird.

In Abbildung 3.12 wird ein mögliches Szenario für eine Abwehr mittels „Interface and Prefix-based Countermeasure“ dargestellt. Zum Zeitpunkt t_1 erhalten die beiden Router A und B über verschiedene Interfaces anfragende Interests von legitimen Consumern und Angreifern. Da die Größe des Interfaces I_1 auf Router B den festgelegten Wert von Threshold = 3 überschreitet, wird der Präfix „/evil“ auf dem Interface I_1 blockiert und ein entsprechendes Alarm-Paket an Router A verschickt.

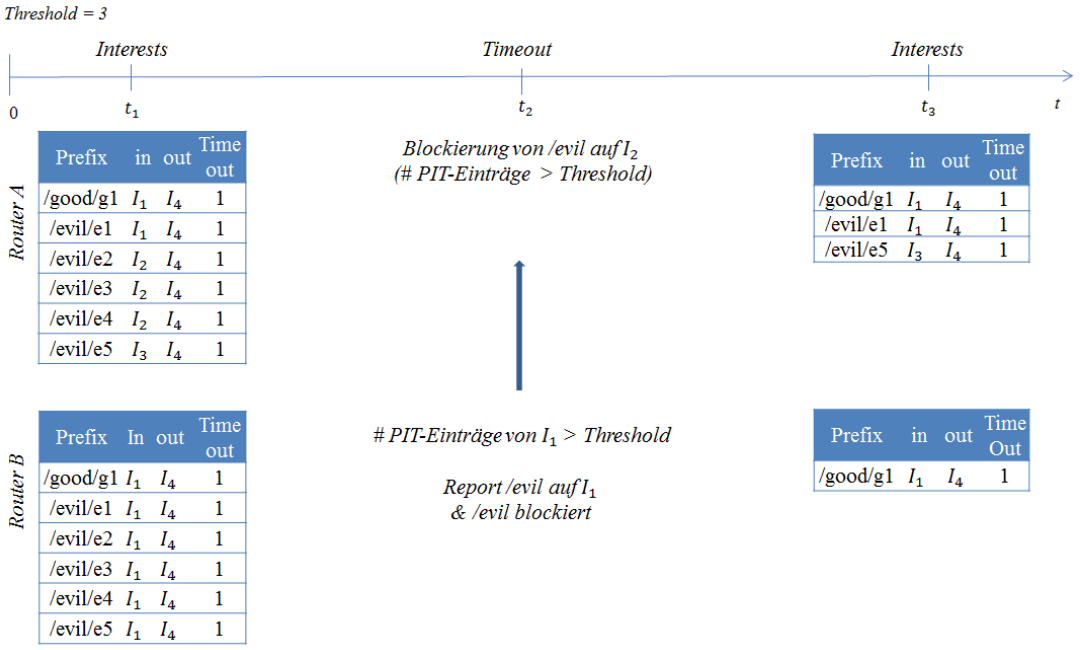

Abbildung 3.12: Zeitstrahl für „Interface and Prefix-based Countermeasure"
PIT-Berechnung für jedes Interface I_i eines Routers

Anschließend blockiert Router A diesen Präfix auf dem Interface I_2, da es das einzige ist, das die Obergrenze überschreitet. Somit werden zum Zeitpunkt t_3 ausschließlich Interests anderer Präfixe auf I_2 und den verbleibenden Interfaces akzeptiert. Da Router B die Interests über nur ein Interface erhält, werden dort anschließend alle Interests mit dem angreifenden Präfix „/evil" verworfen.

3.7 Gegenüberstellung

Um die zuvor beschriebenen Abwehrmethoden zu vergleichen, werden die Bereiche Angriffskriterium, Prüfstelle und Vorgehensweise gegenübergestellt.

Angriffskriterium: Beschreibt die Parameter eines Routers, die in dieser Methode zur Erkennung eines Angriffes genutzt werden.

Prüfstelle: Zeigt auf, welche Teile des Routers für die Feststellung eines Angriffs betrachtet werden: PIT, Interface oder Präfix.

Vorgehensweise: Beschreibt den Methodenablauf, der bei Feststellung eines Angriffs zur Abwehr ausgelöst wird.

Jede Abwehrmethode basiert zunächst auf der Erkennung eines Angriffs. Hierfür sind bestimmte Kriterien definiert. So kann z.B. die gesamte Größe der PIT, der Anteil eines Interfaces oder auch der Anteil eines Präfixes an der PIT betrachtet werden. Aus diesen

Angriffskriterien ergeben sich verschiedene Prüfstellen des Routers: pro Router, pro Interface und pro Präfix (vgl. Tabelle 3.1).

Bei Messung der PIT-Größe pro Router kann zwar das Ausmaß des gesamten Weiterleitungsvorgangs bzw. Angriffs widergespiegelt, jedoch keine weitere Aussage über Ursache bzw. Ursprungsort des Angriffs getroffen werden. Die beiden Methoden „Token Bucket“ und „Resource Allocation“ agieren nach dieser Pro-Router-Variante, wobei sie proaktiv und somit ohne Angriffskriterium funktionieren. Solche Verfahren versuchen, einen Angriff im NDN-Netzwerk nicht wirksam werden zu lassen. Die Interest-Traceback-Methode prüft ebenfalls mit Hilfe des Pro-Router-Verfahrens, wobei dennoch ein Angriffskriterium definiert ist. Neben der üblichen Messung der PIT-Größe wird zusätzlich der PIT-Anstieg gemessen. Da davon ausgegangen wird, dass Angriffe eine stärkere Belastung während eines kleinen Zeitintervalls erreichen, können normale Netzwerkaktivitäten als Nicht-Angriff erkannt werden. Bei Abwehrmethoden, die auf der Pro-Router-Prüfstelle basieren, erfolgt jedoch keine Lokalisierung des Angriffs innerhalb des Routers.

Bei der Messung der PIT-Größe pro Interface kann bei einem Angriff zusätzlich zum gesamten Weiterleitungsvorgang auch das vermeintlich angreifende Interface ermittelt und limitiert werden. Dieses Kriterium wird bei der Poseidon-Local- und der Poseidon-Pushback-Methode genutzt. Um Angriff und normale Netzwerkaktivität unterscheiden zu können, wird hier zusätzlich das Verhältnis von Interests zu Daten-Paketen betrachtet. Bei der Satisfaction-based Interest-acceptance- und Satisfaction-based Pushback-Methode wird ausschließlich das Verhältnis von Interests zu Daten-Paketen für jedes eingehende Interface als Angriffskriterium genutzt, ohne jedoch den PIT-Anteil zu messen. Bei der Pro-Interface-Prüfstelle gibt es keine Möglichkeit, die Lokalisierung des Angriffs für ein Interfaces näher einzugrenzen.

Neben dem Ermitteln der gesamten PIT-Größe und dem Anteil eines Interfaces an der PIT gibt es auch die Möglichkeit, den PIT-Verbrauch pro Präfix zu bestimmen. Dabei kann analog zur Pro-Interface-Variante das Ausmaß der gesamten Weiterleitung festgestellt und exakt ermittelt werden, welcher Präfix den Angriff auf dem Router verursacht. Somit kann der Ursache bzw. dem Ursprungsort dieser Anomalie nachgegangen werden. Dieses Prinzip liegt der Traceback-Methode zu Grunde. Beim TDM-Verfahren gilt ausschließlich das Verhältnis von „gutartigen“ und „bösartigen“ Präfixen als Angriffskriterium. Bei der Pro-Präfix-Prüfstelle gibt es keine Möglichkeit, die Lokalisierung des Angriffs für ein bestimmten Präfix näher einzugrenzen.

Methode	Angriffskriterium	Prüfstelle	Vorgehensweise
Token Bucket with per Interface fairness	—	pro Router	Round Robin
Resource Allocation			PIT Limit setzen
Interest-Traceback	- Gesamt-PIT-Verbrauch - PIT-Anstieg in Intervall		- Spoofed-Paket an alle unbeantw. Interests - Randrouter drosselt Interest-Limit
Poseidon Local	- Anteil des PIT_I zum PIT - Verhältnis von Interests zu Data-Paketen	pro Interface	- Interest–Limit
Poseidon Pushback	- Anteil des PIT_I zum PIT - Verhältnis von Interests zu Data-Paketen		- Interest–Limit - Pushback
Satisfaction-based Interest acceptance	Verhältnis von Interests zu Data-Paketen pro Interface		- Interest Limit je nach Verhältnis - bei 100 % alle verwerfen
Satisfaction-based pushback	Verhältnis von Interests zu Data-Paketen pro Interface		- Interest Limit je nach Verhältnis - bei 100 % alle verwerfen - Pushback
IPC (Interface and Prefix-based Countermeasure)	PIT-Verbrauch jedes Interfaces	pro Interface & pro Präfix	- meist verbrauchender Präfix für das Interface ermittelt - Blockierung und Rückverfolgung durch Alarm-Pakete
DDoS-Abwehrmethode	PIT-Verbrauch eines Präfix	pro Präfix	- alle weiteren Interests verworfen - Router dahinter informiert und Interest Limit drosselt
TDM (threshold-based detecting and mitigating)	Anteil der Art der ankommenden Interests (MI,OI)		- wenn F_S über Threshold je nach Art Kapazität bzw. Rate des Präfixes gedrosselt

Tabelle 3.1: Gegenüberstellung der Prüfstellen pro Abwehrmethode

Die Zuverlässigkeit der einzelnen Techniken hängt maßgeblich von der genutzten Topologie ab, da jede Topologie durch ihre Gegebenheiten signifikante Unterschiede hervorrufen kann. Die Metrik der einzelnen Abwehrmethoden lässt Rückschlüsse auf die geeignete Topologie zu (vgl. Tabelle 3.2).

Die Spalte „Metrik“ beschreibt die Wirkungsweise der Abwehrmethode innerhalb des gesamten NDN-Netzwerks. Es wird zwischen zwei Arten unterschieden: lokal und verteilt. Unter einer lokalen Metrik versteht man eine Abwehrmethode, die ausschließlich auf einem Router agiert, d.h jeder Router im Netzwerk operiert unabhängig. Bei einer verteilten Metrik hingegen, interagieren die Router. Somit kann jede Abwehrmethode anhand ihrer Vorgehensweise diesen zwei Arten zugeordnet werden.In der Spalte „Topologie“ wird eine Aussage dazu getroffen, für welche Netzwerke die jeweilige Methode aufgrund der Funktionsweise geeignet erscheint. So werden Methoden mit einer lokalen Metrik durch ihre Vorgehensweise einer kleinen Topologie zugeschrieben, z.B. „Token Bucket“, „Resource Allocation“ und „Satisfaction-based Interest acceptance“. Das Poseidon-Local- sowie das TDM-Verfahren hingegen werden trotz lokaler Metrik durch ihre Funktionsweise einer kleinen bis mittleren Topologie zugeordnet. Bei verteilten Metriken wird davon ausgegangen, dass diese ihre optimale Wirkungsweise bei mittleren bis größeren Netzwerken erreichen. Die Spalte „Evaluierung“ beschreibt die Leistungswerte, die zur Bewertung der Wirkung einer jeweiligen Abwehrmethode in den jeweiligen Artikeln herangezogen wurden.

Methode	**Metrik**	**Topologie**	**Evaluierung**
Token Bucket with per Interface fairness	lokal	klein	- Verhältnis von Interests zu Daten bei legitimen Consumern (Content Pakete)
Resource Allocation			- Prozentsatz Content Pakete zu Interests - allgemeiner PIT-Verbrauch - allgemeiner Durchsatz
Satisfaction-based Interest acceptance			- Verhältnis von Interests zu Daten bei legitimen Consumern (Content Pakete)
Poseidon Local		klein-mittel	- Prozentsatz Content Pakete zu Interests - allgemeiner PIT-Verbrauch - allgemeiner Durchsatz
TDM			- Kapazität bzw. Rate eines Namespaces
Satisfaction-based pushback	verteilt	mittel - groß	- Verhältnis von Interests zu Daten bei legitimen Consumern (Content Pakete)
Poseidon Distributed			- Prozentsatz Content Pakete zu Interests - allgemeiner PIT-Verbrauch - allgemeiner Durchsatz
Interest-Traceback			- PIT-Verbrauch - Speicherverbrauch - Edge Router: CPU Zyklen (Last)
DDoS-Abwehrmethode			—
IPC			—

Tabelle 3.2: Metrik- und Topologie-Vergleich pro Abwehrmethode

KAPITEL 4

Entwurf des Simulators und Implementierung der Abwehrmethoden

Um die zuvor genannten Abwehrmethoden vergleichen zu können, wird ein NDN-Netzwerk mit Hilfe eines Simulators generiert. Für die vorliegende Arbeit wurde hierfür der ndnSim-Simulator verwendet. Dieser beinhaltet das zuvor erläuterte Kommunikationsmodell von NDN und legt dieses als Modul auf das herkömmliche Kommunikationsmodell, was auf der Architektur des NS-3-Simulators basiert.

4.1 Entwurf des Simulators „ndnSim“

ndnSim ist ein Netzwerk-Layer-Protokoll-Modell, das dem NDN-Kommunikationsmodell entspricht. Es kann neben verschiedenen Link-Layer-Protokollen (CMSA, PPP, 802.11) auch auf den herkömmlichen Netzwerk-Layer-Protokollen (IPv4, IPv6) und Transport-Layer-Protokollen (TCP, UDP) aufgesetzt werden. Dadurch können verschiedene Varianten bei der Entwicklung getestet werden. Der Simulator ist unterteilt in einzelne Module, wobei jedes Modul einem Teil der NDN-Architektur (CS, PIT, FIB) entspricht. Die einzelnen Module des Simulators bestehen aus C++-Klassen.

Design

Das Design von ndnSIM wurde dem der anderen Protokollarten der NS-3-Architektur angepasst. Dabei besitzt jede Komponente des NDN-Kommunikationsmodells ein eigenes Modul, wodurch es bei Veränderungen leichter ausgetauscht werden kann. ndnSim wurde als eigenständiger Protokoll-Stack entwickelt, der auf jedem Knoten eines normalen Netzwerks, was durch NS-3 simuliert wird, genutzt werden kann. Neben dem eigentlichen Protokoll-Stack gibt es noch einige wichtige Nebenfunktionen, wie Applikationen und Helper, die die Simulation und Erzeugung von Szenarien erleichtern.

Im Folgenden werden die wichtigsten Komponenten, die zum Protokoll-Stack gehören, aufgeführt und anschließend einzeln betrachtet.

ndn::L3Protocol	Implementierung des NDN-Protokolls: Senden und Empfangen von Interest- und Data-Paketen über verschiedene Layer durch Interfaces (Faces)
ndn::Face	Implementierung einer einheitlichen Kommunikationsschnittstelle zwischen Applikationen und/oder dem Link-Layer eines Knotens
ndn::ContentStore	Implementierung des NDN-ContentStores als Zwischenspeicher von Data-Paketen
ndn::Pit	Implementierung der NDN-PIT zum Verwalten von anfragenden Interest-Paketen und deren ein- und ausgehenden Faces (in, out)
ndn::Fib	Implementierung der NDN-FIB zum Verwalten von bekannten Datenquellen und deren ausgehenden Faces
ndn::ForwardingStrategy	Implementierung des Weiterleitungsverhaltens von Interest- und Data-Paketen in NDN-Knoten, wodurch beschrieben wird, in welcher Art und Weise die zuvor genannten Datenstrukturen (CS, PIT, FIB) überprüft bzw. genutzt werden.

ndn::L3Protocol

Das Netzwerk-Layer-Protokoll in ndnSim ist die Basis der NDN-Implementierung. Es liegt im Schichtenmodell auf einer Ebene mit den bereits bestehenden Netzwerk-Layer-Protokollen aus dem NS-3-Simulator. Es bildet die Grundlage für jegliche Kommunikation mittels ndn::Face, wobei Applikationen und NDN-Knoten sowohl mit- als auch untereinander kommunizieren können. Das L3Protocol ist für die Weitergabe von eingehenden Paketen zu der verwendeten ForwardingStrategy-Abstraktion durch die Faces zuständig.

ndn::Face

Damit ndnSim im Schichtenmodell flexibel ist, wird das Design unabhängig von der verwendeten bzw. unterliegenden Schicht implementiert und alle Inter-Layer-Kommunikationen über eine Face-Abstraktion verschickt. Somit wird die komplette Kommunikation des Hauptprotokolls, des Netzwerks und der Applikationen über diese Abstraktion geleitet.

Die Face-Abstraktion kann in verschiedenen Arten vorliegen:

- Link-Layer Face (ndn::NetDeviceFace),
- Netwerk-Layer Face (ndn::IPv4Face, ndn::IPv6Face),
- Transport-Layer Face (ndn::TCPFace, ndn:: UDPFace),
- Application-Layer Face (ndn::AppFace).

ndn::ContentStore

Der ContentStore dient in NDN als Zwischenspeicher von präfixbezogenen Daten. Da diese Aufgabe auf unterschiedliche Weise realisiert werden kann, verwendet ndnSim ein Interface,

um verschiedene Implementierungen des ContentStores zu erlauben. Folglich können unterschiedliche Indizierungs- sowie Lookup-Designs, Größenlimitierungen und Ersetzungsregeln für den Cache genutzt werden. Die verwendeteten Implementierungen des ContentStore werden alle durch einen dynamischen Trie-basierten Container mit einer hash-basierten Indizierung der Datennamen organisiert.

Die Abstraktion stellt folgende Funktionen zur Verfügung:

Add:	Es wird ein neues oder existierendes Data-Paket in den Cache geladen.
Lookup:	Die Trie-Datenstruktur wird nach einem zuvor bereits im Cache befindlichen Datennamen durchsucht.

ndn::Pit

Die PIT erfasst jedes an einem NDN-Router eintreffende, anfragende Interest als Eintrag. Ein einzelner PIT-Eintrag trägt in ndnSim folgende Informationen:

- Präfix des Interests,
- Liste aller dazugehörigen ein- sowie ausgehenden Faces,
- Router-Timeout, bestimmte Zeit nachdem der PIT-Eintrag entfernt wird,
- detaillierte Informationen über die Forwarding-Strategy [17].

Neben der herkömmlichen Vorgehensweise der PIT können weitere Regeln festgelegt werden:

Persistent:	Interests über der maximalen Größe werden verworfen.
Random:	Zufällige PIT-Einträge werden bei Erreichen der Limits gelöscht.
LRU (Least-recently-used):	Die ältesten PIT-Einträge mit den wenigsten eingehenden Faces werden bei Erreichen der maximalen Größe entfernt.

Die Implementierung der PIT basiert auf einer Trie-Datenstruktur, wobei eine hash-basierte Indizierung nach Datennamen und Router-Timeout verwendet wird, um die Datenstruktur zu optimieren. Bei einem eingehenden Interest werden die bisherigen PIT-Einträge auf Präfix-Gleichheit geprüft (Longest-Prefix-Match). Falls bereits ein Eintrag existiert, wird die Liste der eingehenden Faces erweitert. Falls kein Eintrag existiert, wird dieser erzeugt. Diese Operationen werden mit Hilfe von folgenden Funktionen durchgeführt:

Lookup:	Findet einen passenden PIT-Eintrag zu einem Daten-Präfix.
Create:	Erzeugt für ein eingehendes Interest einen PIT-Eintrag.
markErased:	Markiert oder löscht einen PIT-Eintrag.
GetSize, Begin, End, Next:	Gibt Größe der PIT aus oder iteriert durch die PIT [17].

ndn::Fib

Die FIB beinhaltet alle für den Router verfügbaren Datenquellen und die dazugehörigen Präfixe. Analog dazu enthält ein FIB-Eintrag einen Daten-Präfix und eine Liste dazugehöriger ausgehender Faces. Die Implementierung der FIB basiert analog zur PIT auf einer Trie-Datenstruktur mit einer hash-basierten Indizierung auf Datenpräfix, wobei jedem Präfix eine geordnete Liste an Faces zur Verfügung steht. Die Reihenfolge der Faces ist abhängig von der Routing-Metrik und dem Data Plane Feedback. [17]

ndn::ForwardingStrategy

Um verschiedene Forwarding-Strategien auf den NDN-Routern agieren zu lassen, wird eine entsprechende Abstraktion (ndn::ForwardingStrategy) verwendet. So können diese genutzt werden, ohne die Module der herkömmlichen Komponenten des Simulators zu verändern. Die Abstraktion ist event-basiert, wodurch es für jede Interaktion des Routers mit einem Interest- oder Data-Paket eine eigenständige Funktion gibt.

Die wichtigen Funktionen lauten:

OnInterest:	Wird bei jedem Eintreffen eines eingehenden Interests ausgeführt.
OnData:	Wird bei Eintreffen eines Data-Pakets durchlaufen.
WillErasePendingInterest:	Ausgeführt, kurz bevor ein PIT-Eintrag entfernt wird.
DidCreatePitentry:	Geworfen, sobald ein PIT-Eintrag erfolgreich erzeugt wurde.
WillSatisfyPendingInterest:	Geworfen, kurz bevor ein anfragendes Interest durch ein passendes Data-Paket erfüllt wird.
CanSendOutInterest:	Prüft, ob ein Interest versendet werden darf oder nicht.
DidSendOutData:	Geworfen, sobald ein Data-Paket erfolgreich gesendet wurde.
TrySendOutInterest:	Geworfen, falls ein Interest in genau diesem Moment gesendet wird.
DidSendOutInterest:	Geworfen, nachdem ein Interest über ein bestimmtes Face erfolgreich versendet wurde.

Da jedes Event, das für eine Weiterleitung notwendig ist, eine eigene Funktion besitzt, kann – je nach Art und Weise – jeder Schritt des Weiterleitungsverhaltens geändert werden. Dadurch ist es möglich, verschiedene Abwehrmethoden als Forwarding-Strategy darzustellen und zu implementieren.

Damit die Abwehrmethoden im Simulator mittels Forwarding-Strategy zum Einsatz kommen, müssen vorerst einige Voraussetzungen erfüllt sein. Es wird eine Topologie benötigt, die verschiedene Knoten eines NDN-Netzwerkes eindeutig festlegt. Dabei ist von Interesse, wo diese in einem virtuellen Koordinatensystem liegen und ob es sich bei den einzelnen Knoten um einen Consumer, einen Router oder einen Producer handelt. Ferner werden die einzelnen Verbindungen der Knoten untereinander und deren Eigenschaften definiert. Dabei wird eine Verbindung durch Quell- und Zielknoten, Bandbreite, Metrik, Delay und Queue organisiert. Die Bandbreite gibt an, welche maximale Datenrate zwischen den Kno-

ten erreicht werden darf. Der Wert der Metrik gibt nähere Auskünfte über die verwendete Routing-Metrik. Delay legt die Zeitverzögerung der Verbindung bei einer Übertragung fest, wodurch diese realistischer simuliert wird. Der Wert der Queue definiert, wie viele Pakete zeitgleich durch diese Verbindung übertragen werden dürfen.

Neben der Topologie muss ein bestimmtes Szenario angelegt sein, das den Verlauf der Simulation eindeutig beschreibt. Es werden weitere Eigenschaften der Knoten, der Verbindungen und der Simulation definiert. Ferner wird festgelegt, welche verschiedenen Applikationen auf den Knoten laufen sollen. So werden auf den Consumern – je nachdem, ob ein legitimer oder ein angreifender Knoten simuliert wird – bestimmte Applikationen ausgeführt. Diese haben Auswirkungen auf die Eigenschaften der Pakete: die berechnete oder angegebene Rate an Interests, den Namen des Interest bzw. den verwendeten Präfix und den LifeTime-Wert des Paketes.

Auf den Producer-Knoten wird eine Applikation installiert, die festlegt, für welche Präfixe diese Daten bereitstehen und welchen Umfang die dazugehörigen Daten haben. Für das Routing wird festgelegt, wohin die einzelnen Präfixe geleitet werden. Da auf den Routern keine Applikation benötigt wird, wird stattdessen definiert, welche Forwarding-Strategy bzw. Abwehrmethode diese benutzen sollen.

Zum besseren Verständnis der Implementierung der Abwehrmethoden werden im Folgenden die zuvor genannten Rahmenbedingungen anhand von Quellcode-Ausschnitten näher erläutert.

Um die erstellte Topologie später in den Simulator einlesen zu können, wird diese und einige weitere Eigenschaften in der Master-Datei angegeben. Bei den Eigenschaften handelt es sich um den Namen der Abwehrmethode, die Anzahl der legitimen und angreifenden Consumer, den Namen des Producers und den Wert der standardmäßigen RoundTripTime, der beschreibt in welcher Zeit ein Paket zum Producer und wieder zurück gelangen kann. Außerdem wird die Anzahl der Simulationsdurchgänge angegeben. Anschließend wird die bereits kompilierte Szenario-Datei mit diesen Informationen als Parameter gestartet, wobei dies als ein Job erfolgt, der in mehrere Threads aufgeteilt und abgearbeitet wird.

In der Szenario-Datei werden die einzelnen Applikationen, die später verwendet werden sollen, initialisiert. Da es zwei verschiedene Applikationen für legitime und angreifende Consumer gibt, wird ein Boolean-Wert benutzt, um dies festzulegen. Anschließend wird der LifeTime-Wert auf den gewünschten Wert gestellt. Außerdem kann die Frequenz der zu verschickenden Interests so gewählt werden, dass sie den empfangenen Daten entspricht. Die Applikation des Producers wird so konfiguriert, dass ausschließlich Daten für den Präfix „/good“ bereitgehalten werden, wobei diese in dem konkreten Fall eine Größe von 1100 Bytes besitzen.

```
1  AppHelper evilAppHelper ("DdosAppMitigate");
2  evilAppHelper.SetAttribute ("Evil", BooleanValue (true));
3  evilAppHelper.SetAttribute ("LifeTime", StringValue ("4s"));
4  evilAppHelper.SetAttribute ("DataBasedLimits", BooleanValue (true));
5  AppHelper goodAppHelper ("DdosAppMitigate");
6  goodAppHelper.SetAttribute ("LifeTime",  StringValue ("4s"));
7  goodAppHelper.SetAttribute ("DataBasedLimits", BooleanValue (true));
8  AppHelper ph ("ns3::ndn::Producer");
9  ph.SetPrefix ("/good");
10 ph.SetAttribute ("PayloadSize", StringValue("1100"));
```

Quellcode 4.1: Szenario-Datei der Interest-Traceback-Methode: Initialisierung

Im nächsten Schritt wird die zuvor erstellte Topologie eingelesen, wodurch die Knoten und die dazugehörigen Verbindungen erzeugt und – je nach legitimer/angreifender Consumer, Router oder Producer – unterteilt werden. Die zu verwendende ForwardingStrategy bzw. Abwehrmethode wird nun auf den bereitgestellten Knoten installiert. Nachdem auf jedem Knoten das Routing-Verfahren installiert und somit jedes Interest durch berechnete Routing-Pfade an den Producer geschickt werden kann, werden die einzelnen Consumer auf die bestimmten Eigenschaften konfiguriert. So versenden legitime Consumer Interests mit dem Präfix „/good“ und angreifende Consumer mit „/evil“. Als Nächstes wird angegeben, wie viele Interests ein legitimer bzw. angreifender Consumer in einer vorgegebenen Zeit verschicken soll. Dies kann durch einen fixen oder berechneten Wert vorgegeben werden. Berechnete Werte sichern eine stabile Netzwerkkommunikation ohne Erscheinungen wie Netzwerk-Stau oder ähnliches.

Abschließend werden die zuvor eingegebenen Einstellungen auf den jeweiligen Consumern und Producern installiert und eine Start- bzw. Endzeit für die Applikation festgelegt.

```
1   for (NodeContainer::Iterator node = goodNodes.Begin (); node != goodNodes.End
        (); node++)
2     {
3       ApplicationContainer goodApp;
4       goodAppHelper.SetPrefix ("/good/"+Names::FindName (*node));
5       goodAppHelper.SetAttribute ("AvgGap", TimeValue (Seconds (1.100 /
            maxNonCongestionShare)));
6       goodApp.Add (goodAppHelper.Install (*node));

8       UniformVariable rand (0, 1);
9       goodApp.Start (Seconds (0.0) + Time::FromDouble (rand.GetValue (), Time::S)
            );
10    }

12  for (NodeContainer::Iterator node = evilNodes.Begin (); node != evilNodes.End
        (); node++)
13    {
14      ApplicationContainer evilApp;
15      evilAppHelper.SetPrefix ("/evil/"+Names::FindName (*node));
16      evilAppHelper.SetAttribute ("AvgGap", TimeValue (Seconds (0.001)));
17      evilApp.Add (evilAppHelper.Install (*node));

19      UniformVariable rand (0, 1);
20      evilApp.Start (Seconds (0.0) + Time::FromDouble (rand.GetValue (), Time::MS
            ));
21      evilApp.Stop (Seconds (900.0) + Time::FromDouble (rand.GetValue (), Time::
            MS));
22    }

24  ph.Install (producerNodes);
```

Quellcode 4.2: Szenario-Datei der Interest-Traceback-Methode: Installation

Durch die Master- und Szenario-Datei wird ein aus der Topologie entstandenes Netzwerk aufgebaut und ein mögliches Szenario für eine Netzwerkkommunikation geschaffen.

Je nach Anzahl der angreifenden Consumer und deren Interest-Frequenz wird dabei ein Interest-Flooding-Angriff auf das NDN-Netzwerk simuliert. Mit Hilfe der auf den Knoten genutzten ForwardingStrategy können die zuvor beschriebenen Abwehrmethoden implementiert und genutzt werden, um den Angriff abzuwehren.

4.2 Implementierung der Abwehrmethoden

Nachdem die Vorgehensweisen der einzelnen Abwehrmethoden bereits erläutert wurden, wird in diesem Abschnitt nun der Fokus auf deren Implementierung gelegt. Dabei werden relevante Auszüge dargestellt und näher erläutert.

Abgesehen von wenigen Ausnahmen, sind die drei Grundbausteine jeder Implementierung:

AnnounceLimits:	Wird in vorgegebenen Zeitintervallen ausgeführt und prüft regelmäßig bestimmte Bedingungen.
OnInterest:	Wird ausschließlich bei Eingang eines Interests ausgeführt.
OnData:	Wird ausschließlich bei Eingang von Daten-Paketen ausgeführt.

Token bucket with per interface fairness

Bei dieser Abwehrmethode handelt es sich um eine der eben genannten Ausnahmen, die die zuvor genannten Funktionen nicht nutzt. Alternativ werden zwei andere essentielle Methoden dargestellt: ProcessFromQueue, TrySendOutInterest.

Bei der ProcessFromQueue-Methode werden alle aktuellen Queues im Router iteriert. Dabei werden pro Queue so viele Interests versendet, wie es das aktuelle Limit vom Router vorgibt. Die Limits werden dabei stetig aktualisiert.

```
1  TokenBucketWithPerInterfaceFairness<Parent>::ProcessFromQueue ()
2  {
3    for (PitQueueMap::iterator queue = m_pitQueues.begin ();
4         queue != m_pitQueues.end ();
5         queue++)
6      {
7        Ptr<Face> outFace = queue->first;
8        Ptr<Limits> faceLimits = outFace->GetObject<Limits> ();
9        while (!queue->second.IsEmpty () && faceLimits->IsBelowLimit ())
10         {
11           Ptr<pit::Entry> pitEntry = queue->second.Pop ();
12           if (pitEntry == 0){break;}

14           pitEntry->OffsetLifetime (Seconds (-0.10) + Seconds (pitEntry->
                 GetInterest ()->GetInterestLifetime ().ToDouble (Time::S)));
15           faceLimits->BorrowLimit ();
16           pitEntry->AddOutgoing (outFace);
17           outFace->SendInterest (pitEntry->GetInterest ());
18           this->DidSendOutInterest (queue->first, outFace, pitEntry->GetInterest
                 (), pitEntry);
19         }
20     }
21 }
```

Quellcode 4.3: Token bucket with per interface fairness: ProcessFromQueue

Durch das Versenden des Interests in Zeile 17 wird die Methode TrySendOutInterest aufgerufen. In der TrySendOutInterest-Methode wird für das aktuelle Interest geprüft, ob es dem Limit-Wert entspricht. Falls dies der Fall ist, wird dieses weitergeleitet und als erfolgreich versendet markiert. Des Weiteren wird der LifeTime-Wert des PIT-Eintrags verändert, damit dieser nicht zu lang in der Queue verbleibt.

```
1  TokenBucketWithPerInterfaceFairness<Parent>::TrySendOutInterest (Ptr<Face> inFace
       ,
2                                                    Ptr<Face> outFace,
3                                                    Ptr<const Interest> interest,
4                                                    Ptr<pit::Entry> pitEntry)
5  {
6    if (pitEntry->GetFwTag<PitQueueTag> () != boost::shared_ptr<PitQueueTag> ())
7      {
8        pitEntry->UpdateLifetime (Seconds (0.10));
9        return true;
10     }

12   if (interest->GetInterestLifetime () < Seconds (0.1)){}

14   pit::Entry::out_iterator outgoing =
15     pitEntry->GetOutgoing ().find (outFace);

17   if (outgoing != pitEntry->GetOutgoing ().end ())
18     {
19       return false;
20     }

22   Ptr<Limits> faceLimits = outFace->GetObject<Limits> ();
23   if (faceLimits->IsBelowLimit ())
24     {
25       faceLimits->BorrowLimit ();
26       pitEntry->AddOutgoing (outFace);
27       outFace->SendInterest (interest);

29       this->DidSendOutInterest (inFace, outFace, interest, pitEntry);
30       return true;
31     }
32   else{}
33   pitEntry->OffsetLifetime (Seconds (-pitEntry->GetInterest ()->
         GetInterestLifetime ().ToDouble (Time::S)));
34   pitEntry->UpdateLifetime (Seconds (0.10));
35   bool enqueued = m_pitQueues[outFace].Enqueue (inFace, pitEntry, 1.0);

37   if (enqueued)
38     {
39       return true;
40     }
41   else
42     return false;
43 }
```

Quellcode 4.4: Token bucket with per interface fairness: TrySendOutInterest

Satisfaction-based Interest Acceptance

Bei der Implementierung dieser Abwehrmethode ist ausschließlich die CanSendOutInterest-Methode von Bedeutung. Anfangs werden für den entsprechenden PIT-Eintrag alle Limits der dazugehörigen FIB-Einträge summiert, um die gesamte Kapazität zu ermitteln. Anschließend wird die aktuelle Wahrscheinlichkeit für das Interest berechnet. Je nach Wert, wird das Interest weitergeleitet oder verworfen.

```
SatisfactionBasedInterestAcceptance<Parent>::CanSendOutInterest (Ptr<Face> inFace
    ,
                                                                 Ptr<Face>
                                                                     outFace,
                                                                 Ptr<const
                                                                     Interest>
                                                                     interest,
                                                                 Ptr<pit::Entry>
                                                                     pitEntry)
{
  for (fib::FaceMetricContainer::type::iterator fibFace = pitEntry->GetFibEntry
      ()->m_faces.begin ();
       fibFace != pitEntry->GetFibEntry ()->m_faces.end ();
       fibFace ++)
    {
      if (!fibFace->GetFace ()->GetObject<Limits> ()->IsEnabled ())
        {
          unlimited = true;
          break;
        }

      totalAllowance += fibFace->GetFace ()->GetObject<Limits> ()->
          GetCurrentLimit ();
    }

  // get unsatisfiedAbs and entire Abs (countAbs)

  if (!unlimited && countAbs > m_graceThreshold * totalAllowance &&
      unsatisfiedAbs > 0)
    {
      double weight = 1.00 - std::min (1.00, unsatisfiedAbs / countAbs);

      double chance = m_acceptChance.GetValue ();
      if (chance > weight)
        {
          return false;
        }
    }

  return super::CanSendOutInterest (inFace, outFace, interest, pitEntry);
}
```

Quellcode 4.5: Satisfaction-based Interest Acceptance: CanSendOutInterest

Satisfaction-based pushback

In der AnnounceLimits-Methode werden anfangs die Gewichte aller Faces errechnet und als Normalisierungswert gespeichert. Nachfolgend wird für jeden FIB-Eintrag ein Interest erstellt und durch alle Faces für jeden datenbringenden Präfix iteriert. Dabei werden erneut die Gewichte für jedes einzelne Face berechnet und anschließend zusammen mit dem entsprechenden Präfix als Interest an das Interface verschickt. Nach einer Sekunde wird die Methode erneut aufgerufen.

```
SatisfactionBasedPushback<Parent>::AnnounceLimits ()
{
  Ptr<L3Protocol> l3 = this->template GetObject<L3Protocol> ();

  for (uint32_t faceId = 0; faceId < l3->GetNFaces (); faceId ++)
    {
```

```
7         Ptr<Face> inFace = l3->GetFace (faceId);
8         // count unsatisfiedAbs and entire abs (countAbs)
9       double weight = 1.00 - std::min (1.00 - m_graceThreshold, unsatisfiedAbs /
            countAbs);
10        sumOfWeights += weight;
11      }
12    for (Ptr<fib::Entry> entry = this->m_fib->Begin ();
13         entry != this->m_fib->End ();
14         entry = this->m_fib->Next (entry))
15      {
16        Ptr<Interest> announceInterest = Create<Interest> ();
17        announceInterest->SetScope (0);

19        for (uint32_t faceId = 0; faceId < l3->GetNFaces (); faceId ++)
20          {
21            Ptr<Face> inFace = l3->GetFace (faceId);
22            const ndnSIM::LoadStatsNode &stats = this->GetStats (Name ());
23            ndnSIM::LoadStatsNode::stats_container::const_iterator item = stats.
                  incoming ().find (inFace);
24            if (item != stats.incoming ().end ())
25              {
26                unsatisfiedAbs = item->second.unsatisfied ().GetStats ().get<0> ();
27                countAbs       = item->second.count ().      GetStats ().get<0> ();
28              }
29            if (countAbs > 0.001)
30              {/*calculate weight and realWeight*/}
31            weight = static_cast<double> (std::max (0.0, weight * totalAllowance));
32                     Ptr<NameComponents> prefixWithLimit = Create<NameComponents>
                           (entry->GetPrefix ());
33            prefixWithLimit->append ("limit").append (boost::lexical_cast<std::
                  string> (weight));
34            announceInterest->SetName (prefixWithLimit);
35            inFace->SendInterest (announceInterest);
36            m_onLimitsAnnounce (inFace, realWeight, weight/totalAllowance, weight);
37          }
38      }
39  Simulator::Schedule (Seconds (1.0), &SatisfactionBasedPushback::AnnounceLimits,
        this);
40  }
```

Quellcode 4.6: Satisfaction-based Pushback: AnnounceLimits

In der OnInterest-Methode werden die Announce-Pakete entgegengenommen und an die ApplyAnnouncedLimit-Methode samt Face übergeben. Reguläre Interests werden weitergeleitet.

```
1  SatisfactionBasedPushback<Parent>::OnInterest (Ptr<Face> face,
2                                                 Ptr<Interest> interest)
3  {
4    if (interest->GetScope () != 0)
5      super::OnInterest (face, interest);
6    else
7      ApplyAnnouncedLimit (face, interest);
8  }
```

Quellcode 4.7: Satisfaction-based Pushback: OnInterest

Da es in dieser Methode keine besonderen Aktionen in der OnData-Methode gibt, wird diese nicht dargestellt. In der ApplyAnnouncedLimit-Methode werden die aktuellen Limit-Werte durch die erhaltenen Announce-Werte ersetzt.

```
SatisfactionBasedPushback<Parent>::ApplyAnnouncedLimit (Ptr<Face> inFace,
                                                        Ptr<const Interest>
                                                            interest)
{
  double limit = boost::lexical_cast<double> (interest->GetName ().get (-1));
  inFace->GetObject<Limits> ()->UpdateCurrentLimit (limit);
}
```

Quellcode 4.8: Satisfaction-based Pushback: ApplyAnnouncedLimit

DoS and DDoS Pushback

Bei der AnnounceLimits-Methode wird regelmäßig geprüft, ob die Größe eines Präfixes (prefixSize) die vorgegebene Limitierung (prefixThreshold) überschreitet. Falls dies der Fall ist, wird der entsprechende PIT-Eintrag beziehungsweise das dazugehörige Interface ermittelt und der angreifende Präfix auf die Liste der zu limitierenden Präfixe gesetzt. Anschließend wird ein Report-Paket mit dem entsprechenden Präfix an das ermittelte Interface geschickt. Alle 0,1 Sekunden ruft die Funktion sich erneut auf.

```
DDoSPushback<Parent>::AnnounceLimits ()
{
Ptr<L3Protocol> l3 = this->template GetObject<L3Protocol> ();
Ptr<Face> face = l3 -> GetFace(0);

for(int i = 0; i < ArraySize(prefixSize); i++)
{
  if(prefixSize[i] > prefixThreshold[i] && Names::FindName (face -> GetNode ()).
      compare(0, 6, "Router") == 0 )
  {
    Ptr<pit::Entry> currentPitEntry = this -> m_pit -> Find(exactEntry[i]);
    ns3::ndn::pit::Entry::in_container inContainer = currentPitEntry ->
        GetIncoming ();
    evilPrefixFace = inContainer.begin () -> m_face;
    evilPrefixes[i] = prefix[i];
    prefixBlacklist[IdToIndex(evilPrefixFace -> GetId ())] = prefix[i];
    Ptr<Data> data = Create<Data> (Create<Packet> (m_virtualPayloadSize));
    data->SetName (Create<Name> ("/Report" + prefix[i]));
      data->SetSignature (239057);
    bool ok = evilPrefixFace->SendData (data);
  }
}
Simulator::Schedule (Seconds (0.1), &DDoSPushback::AnnounceLimits, this);
}
```

Quellcode 4.9: DDoS-Abwehrmethode: AnnounceLimits

Innerhalb der OnInterest-Methode wird nun geprüft, ob das eingehende Interest auf der zu limitierenden Präfix-Liste vermerkt ist. Falls dies der Fall ist, wird das betreffende Interest nicht weitergeleitet, andernfalls wird es an den nächstgelegenen Router gesendet.

```
DDoSPushback<Parent>::OnInterest (Ptr<Face> face,
                                  Ptr<Interest> interest)
{
std::string interestPrefix =  interest -> GetName ().toUri ().substr(0,21);

if(EvilPrefixToIndex(interestPrefix) != 35505)
```

```
7  {/* Interest not forwarded*/ }

9  else
10 {super::OnInterest (face, interest);}
11 }
```

Quellcode 4.10: DDoS-Abwehrmethode: OnInterest

Bei der OnData-Methode wird geprüft, ob es sich um ein reguläres Datenpaket handelt. Falls es ein normales Paket ist, wird dies weitergeleitet. Falls es sich jedoch um ein Report-Paket handelt, wird der angreifende Präfix gespeichert und die PIT auf diesen hin geprüft. Falls dieser enthalten ist, wird auf dem eingehenden Interface das Report-Paket neu erzeugt und an den nächsten Router gesendet.

```
1  DDoSPushback<Parent>::OnData (Ptr<Face> face,
2                                Ptr<Data> data)
3  {
4  uint32_t report = 239057, normal = 0;

6  if(data -> GetSignature () == normal)
7  {super::OnData (face, data);}

9  if(data -> GetSignature () == report && Names::FindName (face -> GetNode ()).
       compare (0, 6, "Router") == 0)
10 {
11   std::string cprefix = data -> GetName ().toUri().substr(7,cprefixSize-6);
12   for(Ptr<pit::Entry> pitEntry = this -> m_pit -> Begin ();
13     (pitEntry != this -> m_pit -> End ()) && (foundPrefix == false);
14     pitEntry = this -> m_pit -> Next (pitEntry))
15     {
16     std::string prefix = pitEntry -> GetPrefix ().toUri().substr(0,21);
17     if(prefix == cprefix)
18     {
19       // transmitted face blacklisted
20     }
21   }
22   if(foundPrefix)
23   {
24     Ptr<Data> data = Create<Data> (Create<Packet> (m_virtualPayloadSize));
25     data->SetName (Create<Name> ("/Report" + cprefix));
26       bool ok = evilFace->SendData (data);
27   }
28 }
29 }
```

Quellcode 4.11: DDoS-Abwehrmethode: OnData

Interest Traceback

In der AnnounceLimits-Methode wird in regelmäßigen Abständen geprüft, ob die PIT-Größe oder die Rate mit der diese steigt, ihre Limits überschreiten. Falls dies der Fall ist, erhält jeder PIT-Eintrag, der zu diesem Zeitpunkt noch existiert, ein Spoofed-Data-Paket. Falls es sich bei dem agierenden Router zusätzlich um einen Randrouter handelt, setzt dieser den Boolean-Wert für den letzten Router im Netzwerk (last) auf true. Daraufhin wird die Methode in 1,0 Sekunden erneut aufgerufen.

```
MitigateTraceback<Parent>::AnnounceLimits ()
{
if(pitSizeInterval1 > pitThreshold || ((pitSizeInterval2-pitSizeInterval1)/2) >
    pitRateThreshold)
{
for (Ptr<pit::Entry> entry = this -> m_pit ->Begin ();
  entry != this -> m_pit ->End ();
  entry = this -> m_pit ->Next (entry))
  {

  ns3::ndn::pit::Entry::in_container inContainer = entry -> GetIncoming ();

  for(ns3::ndn::pit::Entry::in_container::iterator inIterator = inContainer.begin
       ();
      inIterator != inContainer.end ();
        inIterator ++)
    {
    Name evilName = entry -> GetInterest() -> GetName();
        evilFace =  inIterator -> m_face;

    Ptr<Data> data = Create<Data> (Create<Packet> (m_virtualPayloadSize));
    data->SetName (Create<Name> (evilName));
    data->SetSignature(200735);
      bool ok = evilFace->SendData (data);
  }
  if(Names::FindName (face -> GetNode ()).compare(0,6,"Router") == 0)
  {
    if(edge[NameToIndex(Names::FindName (face -> GetNode ()))])
    {
      last[IdToIndex(evilFace -> GetId ())] = true;
    }
  }
}
}
Simulator::Schedule (Seconds (1.0), &MitigateTraceback::AnnounceLimits, this);
}
```

Quellcode 4.12: Interest Traceback: AnnounceLimits

In der OnInterest-Methode wird nun auf diesen Boolean-Wert hin geprüft, so dass eingehende Interests geblockt werden. Falls dies nicht der Fall ist, werden sie regulär weitergeleitet.

```
template<class Parent>
void
MitigateTraceback<Parent>::OnInterest (Ptr<Face> face,
                                       Ptr<Interest> interest)
{
if(last[IdToIndex(face -> GetId ())])
{
  // limit current Face
}
else
{super::OnInterest (face, interest);}
}
```

Quellcode 4.13: Interest Traceback: OnInterest

In der OnData-Methode wird zunächst geprüft, ob es sich bei dem eingehenden Daten-Paket um ein reguläres Paket handelt. Falls dies der Fall ist, wird es weitergeleitet. Handelt es sich jedoch um ein Spoofed-Data-Paket, bei dem es eine vorgegebene Signatur gibt, wird

wiederum geprüft, ob es sich bei diesem Router um den letzten des Netzwerks handelt. Wenn ja, werden durch den Boolean-Wert analog zur Announce-Limit-Methode weitere Interests geblockt. Das Spoofed-Data-Paket wird an den nächsten Router weitergeleitet.

```
1  MitigateTraceback<Parent>::OnData (Ptr<Face> face,
2                                     Ptr<Data> data)
3  {
4  uint32_t spoofed = 200735, ack = 469, normal = 0;

6  if(data -> GetSignature () == normal)
7  {
8    super::OnData (face, data);
9  }
10 if(data -> GetSignature () == spoofed)
11 {
12   if (Names::FindName (face -> GetNode ()).compare (0, 13, "good-Consumer")==0 ||
13       Names::FindName (face -> GetNode ()).compare (0, 13, "evil-Consumer")==0)
14   {super::OnData (face,data);}
15   else
16   {
17     if(edge[NameToIndex(Names::FindName (face -> GetNode ()))])
18     {
19       last[IdToIndex(face -> GetId ())] = true;
20     }

22     super::OnData (face, data);
23   }
24 }
25 }
```

Quellcode 4.14: Interest Traceback: OnData

Resource Allocation

Da in dieser Abwehrmethode ausschließlich eine PIT-Obergrenze gesetzt wird, wird nur die OnInterest-Methode verändert.

Bei Eingang eines Interests wird geprüft, ob die aktuelle PIT-Größe innerhalb der zulässigen Größe liegt, in diesem Fall 1200 Einträge. Wenn ja, wird das eingehende Interest ordnungsgemäß weitergeleitet. Falls nicht, wird es verworfen.

```
1  PoseidonResourceAllocation<Parent>::OnInterest (Ptr<Face> face,
2                                                  Ptr<Interest> interest)
3  {
4    if(this -> m_pit -> GetSize() >= 1200){}
5    else
6    {
7      super::OnInterest (face, interest);
8    }
9  }
```

Quellcode 4.15: Resource Allocation: OnInterest

Poseidon Local

In der AnnounceLimits-Methode werden alle existierenden Interfaces daraufhin geprüft, ob die Werte $\omega(r_i^j, t_k), \rho(r_i^j, t_k)$ die gegebenen Limits $\Omega(r_i^j)$ und $P(r_i^j)$ überschreiten. Falls dies der Fall ist, werden die beiden Limit-Werte reduziert und das aktuelle Interface als limitiert markiert. Falls dies jedoch nicht der Fall ist, wird geprüft, ob das Interface bereits limitiert war und beide Werte wieder unterhalb der Limits liegen. Wenn ja, werden die Limit-Wert etwas erhöht. Daraufhin wird die Methode in 0,1 Sekunden erneut aufgerufen.

```
1  PoseidonPushbackLocal<Parent>::AnnounceLimits ()
2  {
3  Ptr<L3Protocol> l3 = this->template GetObject<L3Protocol> ();

5  if(Names::FindName (l3 -> GetFace (0) -> GetNode ()).compare(0, 6, "Router") ==
       0)
6  {

8    for (uint32_t i = 0;  i < ArraySize(FaceID); i ++)
9    {
10   int currentomegaSize = IntervaloSize(currentIndex);
11   int currentpSize =  pSizeInt2[currentIndex] - pSizeInt1[currentIndex];

13   if(currentomegaSize > omegaThreshold[currentIndex] && currentpSize >
         peThreshold[currentIndex])
14   {
15     omegaThreshold[currentIndex] = omegaThreshold[currentIndex] / scale;
16     peThreshold[currentIndex] = peThreshold[currentIndex] / scale;
17     limitedFace[currentIndex] = true;
18   }
19   else
20   {
21     if (limitedFace[currentIndex])
22     {
23       if(currentomegaSize < omegaThreshold[currentIndex] && currentpSize <
             peThreshold[currentIndex])
24       {
25         omegaThreshold[currentIndex] = omegaThreshold[currentIndex] + (
               omegaThreshold[currentIndex]/8);
26         peThreshold[currentIndex] = peThreshold[currentIndex] + (peThreshold[
               currentIndex]/8);
27       }
28     }
29   }
30   }
31 }

33 Simulator::Schedule (Seconds (0.1), &PoseidonPushbackLocal::AnnounceLimits, this)
       ;
34 }
```

Quellcode 4.16: Poseidon Local: AnnounceLimits

Bei der OnInterest-Methode wird geprüft, ob die Werte des Interfaces berechnet werden müssen. Falls dies der Fall ist, werden diese mit Hilfe der IntervalpSize-Funktion und des Arrays „incomeInterest“ erstmalig berechnet, nach 60 ms erneut und dann gespeichert. Falls das Interface bereits als limitiert markiert ist, werden eingehende Interests blockiert und das Interface so gedrosselt. Andernfalls werden die Interests regulär weitergeleitet.

```
PoseidonPushbackLocal<Parent>::OnInterest (Ptr<Face> face,
                                           Ptr<Interest> interest)
{
int oiIndex = IdToIndex(face -> GetId ());
incomeInterest[oiIndex]++;

if(measureInterests[oiIndex])
{
  double currentTime = Simulator::Now ().ToDouble (Time::S);

  if(firstMeasureInterest[oiIndex])
  {
    interestInt1[oiIndex] = incomeInterest[oiIndex];
    pSizeInt1[oiIndex] = IntervalpSize(oiIndex);
    startInterest = Simulator::Now ().ToDouble (Time::S);
    firstMeasureInterest[oiIndex] = false;
  }
  if(currentTime > (startInterest + 0.06))
  {
    pSizeInt2[oiIndex] = IntervalpSize(oiIndex);
    interestInt2[oiIndex] = incomeInterest[oiIndex];
    measureInterests[oiIndex] = false;
    firstMeasureInterest[oiIndex] = true;
  }
}
if(limitedFace[oiIndex])
{
  // limit current Face
}
else
{
super::OnInterest (face, interest);
}
}
```

Quellcode 4.17: Poseidon Local: OnInterest

In der OnData-Methode werden die verbleibenden Parameter, die zur Berechnung der Werte $\omega(r_i^j, t_k), \rho(r_i^j, t_k)$ benötigt werden, ermittelt. Dies geschieht analog zur Vorgehensweise, die bei OnInterest gebraucht wurde. Handelt es sich bei dem eingehenden Daten-Paket um ein reguläres Paket, wird dieses weitergeleitet.

```
PoseidonPushbackLocal<Parent>::OnData (Ptr<Face> face,
                      Ptr<Data> data)
{
int odIndex = IdToIndex(face -> GetId ());
incomeData[odIndex]++;

if(measureData[odIndex])
{
  double currentTime = Simulator::Now ().ToDouble (Time::S);

  if(firstMeasureData[odIndex])
  {
    dataInt1[odIndex] = incomeData[odIndex];
    startData = Simulator::Now ().ToDouble (Time::S);
    firstMeasureData[odIndex] = false;
  }

  if(currentTime > (startData + 0.06))
  {
    dataInt2[odIndex] = incomeData[odIndex];
    measureData[odIndex] = false;
```

```
22      firstMeasureData[odIndex] = true;
23    }
24  }
25  if (data -> GetSignature () == 0)
26  {
27        super::OnData (face, data);
28  }
29  }
```

Quellcode 4.18: Poseidon Local: OnData

Poseidon Distributed

In der AnnounceLimits-Methode werden analog zum Poseidon-Local-Verfahren die Werte jedes Interface auf Überschreitung der Limits geprüft. Falls dies der Fall ist und der letzte Alarm bereits vor mehr als 60 *ms* (waittime) stattgefunden hat, wird dieses Interface als limitiert markiert und ein Alert-Paket mit dem entsprechenden Präfix (evilPrefix) an den nächsten Router geschickt. Wenn das Interface bereits limitiert ist und die beiden Werte ihre Limits nun nicht mehr überschreiten, werden die Limits erhöht. Nach 0,1s wird die Methode erneut aufgerufen.

```
1   PoseidonPushbackDistributed<Parent>::AnnounceLimits ()
2   {
3   if(Names::FindName (13 -> GetFace (0) -> GetNode ()).compare(0, 6, "Router") ==
        0)
4   {
5     for (uint32_t i = 0; i < ArraySize(faceID); i ++)
6     {
7       int currentoSize = IntervaloSize(currentIndex);
8     int currentpSize =  pitSizeInt2[currentIndex] - pitSizeInt1[currentIndex];

10      if(currentoSize > oThreshold[currentIndex] && currentpSize > pThreshold[
            currentIndex])
11      {
12          if(lastAlert < (Seconds(Simulator::Now ().ToDouble (Time::S) - wait_time)
                ))
13          {
14        // reduce oThreshold and pThreshold
15        limited[currentIndex] = true;
16        Ptr<Data> data = Create<Data> (Create<Packet> (m_virtualPayloadSize));
17        std::string evilPrefix = limitedPrefix[currentIndex];
18        data->SetName (Create<Name> ("/pushback/alerts" + evilPrefix));
19            data->SetTimestamp (Simulator::Now ());
20            data->SetFreshness (Seconds(0));
21            data->SetSignature (12345);
22            bool ok = inFace->SendData (data);
23          }
24    }
25    else
26      {
27          if (limited[currentIndex])
28          {
29              if(currentoSize < oThreshold[currentIndex] && currentpSize <
                    pThreshold[currentIndex])
30              {
31                  // increase othreshold and pThreshold
32              }
33          }
34      }
35    }
```

```
36 }
37 Simulator::Schedule (Seconds (0.1), &PoseidonPushbackDistributed::AnnounceLimits,
       this);
38 }
```

Quellcode 4.19: Poseidon Distributed: AnnounceLimits

Die OnInterest-Methode verhält sich identisch zu Poseidon Local.

```
1  PoseidonPushbackDistributed<Parent>::OnInterest (Ptr<Face> face,
2                                        Ptr<Interest> interest)
3  {
4    if(measureInterestsDis[oiIndex])
5    {
6      double currentTime = Simulator::Now ().ToDouble (Time::S);
7      if(firstMeasureInterestDis[oiIndex])
8      {
9        incomeInterestInt1[oiIndex] = incomingInterest[oiIndex];
10       pitSizeInt1[oiIndex] = IntervalpSize(oiIndex);
11       startInterestDis = Simulator::Now ().ToDouble (Time::S);
12       firstMeasureInterestDis[oiIndex] = false;
13     }
14     if(currentTime > (startInterestDis + 0.06))
15     {
16       pitSizeInt2[oiIndex] = IntervalpSize(oiIndex);
17       incomeInterestInt2[oiIndex] = incomingInterest[oiIndex];
18       measureInterestsDis[oiIndex] = false;
19       firstMeasureInterestDis[oiIndex] = true;
20     }
21   }
22     if(limited[oiIndex])
23     {
24     //limit current Face
25     }
26     else
27     {
28     super::OnInterest (face, interest);
29     }
30 }
```

Quellcode 4.20: Poseidon Distributed: OnInterest

Bei OnData wird geprüft, ob es sich bei dem eingehenden Daten-Paket um ein Alert-Paket (Signature = 12345) handelt und der letzte Alarm bereits vor mehr als 60 *ms* (waittime) stattgefunden hat. Falls dies der Fall ist, werden die Limits des Interface mit dem entsprechenden Präfix reduziert. Handelt es sich jedoch um ein reguläres Paket (Signature = 0), werden die Werte analog zur Poseidon-Local-Variante berechnet und das Daten-Paket weitergeleitet.

```
1  PoseidonPushbackDistributed<Parent>::OnData (Ptr<Face> face,
2                                          Ptr<Data> data)
3  {
4  if (data -> GetSignature () == 12345 && lastAlert < (Simulator::Now ().ToDouble (
       Time::S) - wait_time
5      && Names::FindName (face -> GetNode ()).compare(0 ,6 ,"Router") == 0  ))
6  {
7    for(Ptr<pit::Entry> pitEntry = this -> m_pit->Begin ();
8          pitEntry != this -> m_pit->End ();
9          pitEntry = this -> m_pit->Next (pitEntry))
10     {
```

```
11      if(pitPrefix == currentPrefix)
12      {
13        //get the index of the evilFace
14      }
15    }
16    oThreshold[odIndex] = oThreshold[odIndex] / scaling;
17      pThreshold[odIndex] = pThreshold[odIndex] / scaling;
18  }
19  if (data -> GetSignature () == 12345 && lastAlert >= Seconds(Simulator::Now ().
        ToDouble (Time::S)) - wait_time ){}
20  if (data -> GetSignature () == 0)
21  {
22    if(measureDataDis[nodIndex])
23    {
24      double currentTime = Simulator::Now ().ToDouble (Time::S);
25      if(firstMeasureDataDis[nodIndex])
26      {
27        incomeDataInt1[nodIndex] = incomingData[nodIndex];
28        startDataDis = Simulator::Now ().ToDouble (Time::S);
29        firstMeasureDataDis[nodIndex] = false;
30      }
31      if(currentTime > (startDataDis + 0.06))
32      {
33        incomeDataInt2[nodIndex] = incomingData[nodIndex];
34        measureDataDis[nodIndex] = false;
35      }
36    }
37    super::OnData (face, data);
38  }
39  }
```

Quellcode 4.21: Poseidon Distributed: OnData

Threshold-based detecting and mitigating (TDM)

Da in dieser Abwehrmethode alle Funktionen event-basiert sind, gibt es keine Funktionen, die in der AnnounceLimits-Methode laufen. In der WillEraseTimeOutPendingInterest-Methode, die jedes Mal aufgerufen wird, sobald ein PIT-Eintrag abläuft, werden zwei Boolean-Werte gesetzt. Diese definieren, ob der letzte PIT-Eintrag zu einem entsprechenden Präfix abgelaufen (erased) ist oder durch Daten erfüllt (satisfied) wurde. Dementsprechend wird der Boolean-Wert auf „abgelaufen“ gesetzt.

```
1  TDM<Parent>::WillEraseTimedOutPendingInterest (Ptr<pit::Entry> pitEntry)
2  {
3    int currentPrefixIndex = PrefixToIndex(currentPrefix);
4    satisfied[currentPrefixIndex] = false;
5    erased[currentPrefixIndex] = true;

7    super::WillEraseTimedOutPendingInterest (pitEntry);
8  }
```

Quellcode 4.22: TDM: WillEraseTimeOutPendingInterest

In der OnInterest-Methode wird geprüft, ob der eingehende Präfix bereits limitiert ist. Falls dies der Fall ist, wird der *Capacity*-Wert reduziert und der entsprechende Präfix weiter gedrosselt. Andernfalls wird geprüft, ob der letzte PIT-Eintrag, der zu diesem Präfix gehört, erfüllt wurde oder abgelaufen ist. Falls er erfüllt wurde, werden die einzelnen Zähler

des TDM-Verfahrens gesetzt und die Limitierung des Präfixes aufgehoben. Wenn der PIT-Eintrag abgelaufen ist, werden die Zähler entsprechend gesetzt, die Limitierung aktiviert und das Interest weitergeleitet.

```
1  template<class Parent>
2  void
3  TDM<Parent>::OnInterest (Ptr<Face> face,
4                           Ptr<Interest> interest)
5  {
6  int currentPrefixIndex = PrefixToIndex(currentPrefix);

8  if(limited[currentPrefixIndex])
9  {
10   capacity[currentPrefixIndex] = 0.5 * capacity[currentPrefixIndex];
11   // limit current Face to capacity
12 }
13 else
14 {
15     if(satisfied[currentPrefixIndex] == true && erased[currentPrefixIndex] ==
           false)
16     {
17         if(mode[currentPrefixIndex] == true)
18         {
19               ci[currentPrefixIndex] = 1;
20               mode[currentPrefixIndex] = false;
21         }
22         else
23         {
24               ci[currentPrefixIndex]++;
25         }
26     if(ci[currentPrefixIndex] >= cth[currentPrefixIndex])
27     {
28       fs[currentPrefixIndex]++;
29       if(fs[currentPrefixIndex] >= fth[currentPrefixIndex])
30       {
31         limited[currentPrefixIndex] = false;
32       }
33     }
34   }
35   else if (satisfied[currentPrefixIndex] == false && erased[currentPrefixIndex]
         == true)
36     {
37         if(mode[currentPrefixIndex] == false)
38         {
39           ci[currentPrefixIndex] = 1;
40             mode[currentPrefixIndex] = true;
41         }
42     else
43         {
44           ci[currentPrefixIndex]++;
45         }

47     if(ci[currentPrefixIndex] >= cth[currentPrefixIndex])
48     {
49       fs[currentPrefixIndex] = 0;
50       limited[currentPrefixIndex] = true;
51     }
52     }

54     super::OnInterest (face, interest);
55 }
56 }
```

Quellcode 4.23: TDM: OnInterest

In der OnData-Methode werden analog zur WillEraseTimeOutPendingInterest-Methode die beiden zuvor genannten Boolean-Werte gesetzt. Hierbei jedoch so, dass der entsprechende Präfix durch die eingehenden Daten erfüllt wurde.

```
template<class Parent>
void
    TDM<Parent>::OnData (Ptr<Face> face,
                         Ptr<Data> data)
{
  int currentPrefixIndex = PrefixToIndex(currentPrefix);
  erased[currentPrefixIndex] = false;
  satisfied[currentPrefixIndex] = true;

  super::OnData (face, data);
}
```

Quellcode 4.24: TDM: OnData

Interface and Prefix-based Countermeasure (IPC)

In der AnnounceLimits-Methode des IPC-Verfahrens wird für jedes Interface der jeweilige Präfix mit den meisten PIT-Einträgen durch die GetMaxPrefixes-Methode ermittelt. Anschließend wird für jedes Interface geprüft, ob es den Wert der oberen Grenze von 500 PIT-Einträgen überschreitet. Falls dies der Fall ist, wird der Präfix und das Interface auf eine entsprechende Blacklist gesetzt und anschließend ein modifiziertes Alarm-Paket an den entsprechenden Router gesendet. Diese Funktion wird alle 0,5 Sekunden erneut aufgerufen.

```
IPM<Parent>::AnnounceLimits ()
{
  if(Names::FindName (face -> GetNode ()).compare (0, 6, "Router") == 0)
  {
    GetMaxPrefixes();
    for (int i = 0; i < ArraySize(faceID); i ++)
    {
      if(FaceSize[i] > 500)
      {
      maxPrefix = maxPrefixe[i];
      evilFaces[evilIt] = faceID[i];
      evilPrefixes[evilIt] = maxPrefix;
      evilIt++;

        Ptr<Data> data = Create<Data> (Create<Packet> (m_virtualPayloadSize));
      data->SetName (Create<Name> ("/IPMReport" + maxPrefix));
          data->SetTimestamp (Simulator::Now ());
      data->SetFreshness (Seconds(0));
          data->SetSignature (539047);
      Cface[i]->SendData (data);
      }
    }
    }
    Simulator::Schedule (Seconds (0.5), &IPM::AnnounceLimits, this);
}
```

Quellcode 4.25: IPC: AnnounceLimits

In der OnInterest-Methode wird anfangs der verwendete Präfix des Interests zwischengespeichert und anschließend die Blacklist auf diesen Präfix und das dazugehörige Interface hin untersucht. Falls diese dort enthalten sind, wird das Interest verworfen, wenn nicht, wird es weitergeleitet.

```
IPM<Parent>::OnInterest (Ptr<Face> face,
                         Ptr<Interest> interest)
{
  currentPrefix = interest -> GetName().toUri().substr(0,21);

    for (int i = 0; i < ArraySize(evilPrefixes) && found == false; i ++)
    {
    if(evilPrefixes[i] == currentPrefix)
    {
    if(evilFaces[i] == face -> GetId())
    {
      evilPrefix = currentPrefix;
      found = true;
      foundFace = true;
    }
    }
    }
    if(evilPrefix != "" && foundFace){/*drop*/}
    else
    {
  super::OnInterest (face, interest);
    }
}
```

Quellcode 4.26: IPC: OnInterest

In der OnData-Methode wird geprüft, ob es sich bei dem ankommenden Datenpaket um ein Alarm-Paket handelt. Falls dies der Fall ist, wird ein entsprechender PIT-Eintrag ermittelt und das entsprechende Interface bei Überschreiten der Obergrenze zwischengespeichert. Nachfolgend wird der Präfix und das Interface auf die entsprechende Blacklist gesetzt und das Alarm-Paket an den nächsten Router geschickt. Wenn es sich beim dem Daten-Paket nicht um ein Alarm-Paket handelt, wird dieses regulär weitergeleitet.

```
IPM<Parent>::OnData (Ptr<Face> face,
                     Ptr<Data> data)
{
  icpreport = 539047;
  if(data -> GetSignature () == icpreport && Names::FindName (face -> GetNode ())
      .compare (0, 6, "Router") == 0)
  {
  std::string currentPrefix = data -> GetName ().toUri().substr(10,21) + "/";

  for(Ptr<pit::Entry> pitEntry = this -> m_pit -> Begin ();
           pitEntry != this -> m_pit -> End () && foundPrefix == false;
            pitEntry = this -> m_pit -> Next (pitEntry))
       {
    std::string pitPrefix = pitEntry -> GetPrefix ().toUri().substr(0,21);
    if(pitPrefix == currentPrefix)
    {
      foundPrefix = true;
      ns3::ndn::pit::Entry::in_container inContainerPrefix = pitEntry ->
          GetIncoming ();
      evilFace = inContainerPrefix.begin () -> m_face;
      if(FaceSize[FaceToIndex(evilFace -> GetId())] > 500)
      {
```

```
21          evilFaces[evilIt] = evilFace -> GetId();
22          evilPrefixes[evilIt] = currentPrefix;
23          evilIt++;
24        }
25      }
26    }

28    if(foundPrefix)
29    {
30      Ptr<Data> data = Create<Data> (Create<Packet> (m_virtualPayloadSize));
31      data->SetName (Create<Name> ("/IPMReport" + currentPrefix));
32      data->SetFreshness (Seconds(0));
33          data->SetSignature (539047);
34          evilFace->SendData (data);

36    }
37    }
38    else{
39        super::OnData (face, data);
40    }
41  }
```

Quellcode 4.27: IPC: OnData

KAPITEL 5

Verifikation der Simulationen

Da für sechs der zuvor genannten Abwehrmethoden für den verwendeten Simulator ndnSim keinerlei Implementierungen vorhanden waren, mussten diese zunächst eigenständig vorgenommen werden. Für drei Abwehrmethoden aus dem Artikel „Interest Flooding Attack and Countermeasures in Named Data Networking“ gab es bereits Implementierungen.

Um die Funktionsweise bzw. Implementierung jeder Abwehrmethode auf Richtigkeit bezüglich des referenzierten Originals prüfen zu können, bedurfte es bereits vorliegender Messwerte für einen vergleichbaren Simulator. Diese lagen für sechs Abwehrmethoden vor. Dementsprechend wurde die Verifikation für die Implementierung dieser Methoden durchgeführt.

Um mögliche Nebeneffekte zu vermeiden, wurden die in den Artikeln beschriebenen Szenarien und Topologien übernommen. Um sicher zustellen, dass es sich bei den zuvor gezeigten Implementierungen um äquivalente Umsetzungen der beschriebenen Abwehrmethoden handelt, werden die in Diagrammen ausgewerteten Messwerte der folgenden Artikel mit denen der eigenen Implementierungen visuell verglichen.

5.1 Interest Flooding Attack and Countermeasures in Named Data Networking

Da es sich hierbei um Abwehrmethoden der Entwickler des verwendeten Simulators ndnSim handelt, waren die Implementierungen bereits vorhanden und mussten nicht erneut programmiert werden.

Um die Simulationen zu verifizieren, werden nachfolgend Messwert-Abbildungen aus dem Artikel (originale Implementierung) entsprechenden eigenen Diagrammen gegenübergestellt (eigene Implementierung).

Die Abbildung 5.1 bzw. 5.2 geben den Verlauf des Verhältnisses von Interests zu Daten über ein bestimmtes Zeitintervall während eines Interest-Flooding-Angriffs wieder.

Der visuelle Vergleich der einzelnen Abwehrmethoden zeigt, dass es eine Übereinstimmung der Messwerte bei den Methoden, „Token bucket with per interface fairness“ (rot) und

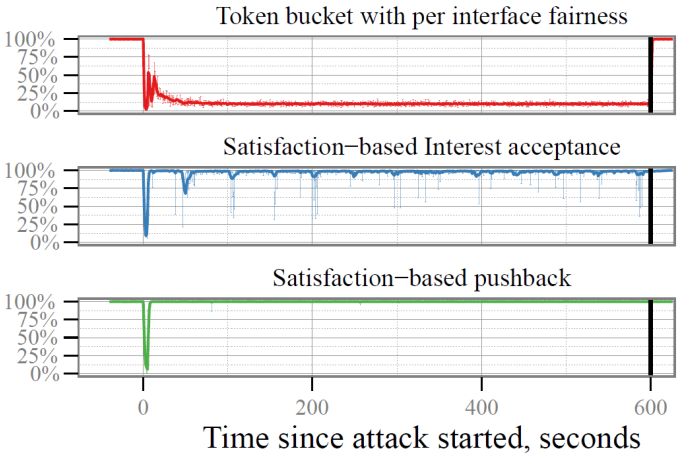

Abbildung 5.1: Interests-Daten-Verhältnis 1 (originale Implementierung) [2]

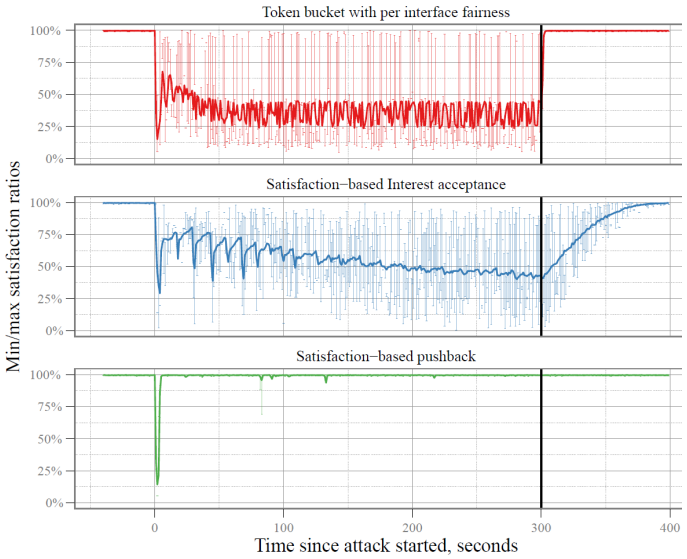

Abbildung 5.2: Interests-Daten-Verhältnis 1 (eigene Implementierung)

„Satisfaction-based pushback“ (grün), gibt. Bei dem Pushback-Verfahren bleiben die Werte über die Zeit konstant bei 100%. Die Token-Bucket-Methode hält im Verlauf des Angriffs eine datenbringende-Interest-Rate von ca. 20%, wobei jedoch auffällt, dass der Wer bei der eigenen Implementierung etwas höher ist.

Bei der „Satisfaction-based Interest acceptance“-Methode (blau) kommt es zu leichten Abweichungen. Die eigene Implementierung weist niedrigere Werte als die originale Implementierung auf. Da jedoch alle Abwehrmethoden unter gleichen Bedingungen simuliert wurden und die zuvor genannten Graphen übereinstimmten, wurde bezüglich der Abweichung bei den Autoren nachgefragt. Diese schlossen eine abweichende Verlaufskurve durch inzwischen erfolgte Änderungen nicht aus.

Neben der Abbildung zum Verlauf des Verhältnisses von Interests zu Daten der Consumer über ein bestimmtes Zeitintervall, wurde eine weitere Darstellung gewählt, die zwar ebenfalls das Verhältnis von Interests zu Daten – allerdings bei verschiedenen Varianten der Angreifer-Anzahl zur legitimen-User-Anzahl – anzeigt. Die farbliche Zuordnung der einzelnen Abwehrmethoden bleibt gleich. Während Abbildung 5.3 das Original darstellt, zeigt die Abbildung 5.4 die eigenständig erstellte Auswertung der Implementierung.

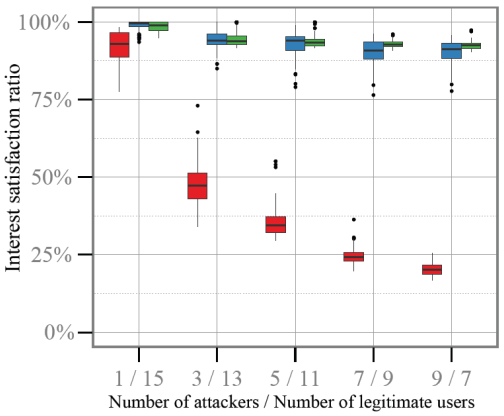

Abbildung 5.3: Interests-Daten-Verhältnis 2 (originale Implementierung) [2]

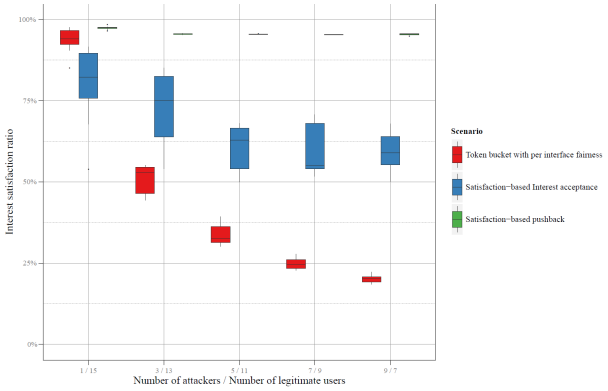

Abbildung 5.4: Interests-Daten-Verhältnis 2 (eigene Implementierung)

Analog zur vorhergehenden Darstellung fällt auf, dass die Methoden „Token bucket with per interface fairness“ und „Satisfaction-based pushback“ in beiden Grafiken sehr ähnliche Messwerte aufweisen. Die roten Balken zeigen, dass das Token-Bucket-Verfahren mit zunehmender Angreifer-Anzahl an Daten verliert. Das Satisfaction-Pushback-Verfahren hingehen, hält trotz steigender Angreifer-Anzahl die Datenverteilung während des Angriffs nahezu bei 100%.

Bei der Satisfaction-based-Interest-acceptance-Methode unterscheiden sich die Werte der Abwehrmethode analog zum vorhergehenden Vergleich. In der Abbildung 5.3 bleiben die Werte der datenbringenden Interests ähnlich der Pushback-Methode nahezu bei 100 %. In der Abbildung 5.4 hingegen nehmen die Werte mit zunehmender Angreifer-Anzahl ab und stagnieren bei 60%.

5.2 Mitigate DDoS Attacks in NDN by Interest Traceback

In diesem Artikel wird ausschließlich die Methode “Interest Traceback“ vorgestellt. Die dortige Darstellung der Messwerte soll mit der eigenen Implementierung verglichen werden, wobei in beiden Fällen das gleiche Szenario genutzt wird. Da jedoch die angegebene Topologie etwas kleiner ausfällt, als es zur Zeit der Publikation der Fall war, wurde das Szenario entsprechend angepasst.

Abbildung 5.6 zeigt die absolute PIT-Größe aller Router im Verlauf eines Zeitintervalls. Diese Größe wurde in PIT-Einträgen pro Sekunde für alle Router summiert. Dafür wurde ein Interest-Flooding-Angriff für eine Zeitdauer von 28 Sekunden simuliert, wobei die

Abwehrmethode nach exakt 12 Sekunden gestartet wird. Dadurch kann die Wirkungsweise der Methode bei größtmöglicher Angriffslast getestet werden. Dieses Szenario wurde für drei verschiedene Interest-LifeTime-Werte: 4 Sekunden, 2 Sekunden, 1 Sekunde wiederholt.

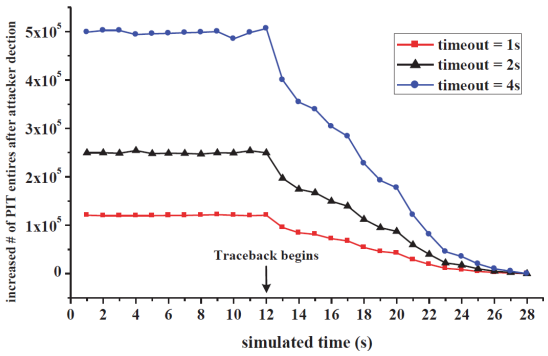

Abbildung 5.5: Absolute PIT-Größe (originale Implementierung) [3]

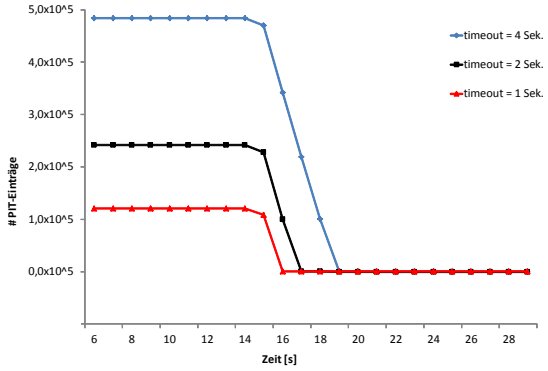

Abbildung 5.6: Absolute PIT-Größe (eigene Implementierung)

Bei einem direkten Vergleich der beiden Abbildungen 5.6 und 5.5 fällt auf, dass diese sich stark ähneln. Der Graph des 4-Sekunden-Timeouts beginnt in beiden Fällen bei rund $5 \cdot 10^5$ PIT-Einträgen im Zeitintervall $[0, 1]$. Dieser Wert bleibt in beiden Implementierungen bis zur Aktivierung der Abwehrmethode recht konstant. Von dort an fallen die Messwerte mit fortlaufendem Zeitintervall bis zum Wert 0, wobei jedoch in der eigenen Variante ein schnellerer Abfall der PIT-Einträge als bei der originalen Methode zu verzeichnen ist. Es kann davon ausgegangen werden, dass dieser Effekt durch die Verwendung verschiedener Simulatoren entsteht. Die beiden anderen Messreihen mit einem 2- bzw. 1-Sekunden-Timout verhalten sich ähnlich, starten jedoch in beiden Fällen bei rund $2 \cdot 10^5$ und $1 \cdot 10^5$ PIT-Einträgen.

Trotz des leicht unterschiedlichen Abfalls der Messwerte nach Aktivierung der Abwehrmethoden, können diese Methoden als äquivalent agierend betrachtet werden.

5.3 Poseidon: Mitigating Interest Flooding DDoS Attacks in Named Data Networking

In diesem Artikel werden die drei Abwehrmethoden „Resource Allocation“, „Poseidon Local“ und „Poseidon Distributed“ vorgestellt. Bei dem Resource-Allocation-Verfahren handelt es sich um eine Abwehrmethode, die ausschließlich auf bestimmten Parametern basiert, so dass kein direkter Vergleich nötig ist.

Bei den Poseidon-Local- und Poseidon-Distributed-Verfahren hingegen muss ein direkter Messwert-Vergleich zum Abgleich beider Implementierungen herangezogen werden. Um Nebeneffekte zu minimieren, wurde das gleiche Szenario sowie die Topologie verwendet, wobei jedoch einige Parameter aufgrund von Simulator-Unterschieden entsprechend angepasst wurden.

Poseidon Local und Poseidon Distributed

Bei der Poseidon-Local-Methode wurde für die Verifikation eine Abbildung gewählt, die die PIT-Größe bestimmter, für die verwendete Topologie entscheidende Router in einem vorgegebenen Zeitintervall darstellt. Diese Router liegen vorwiegend in der näheren Umgebung des Producers, da dort das Ausmaß des Angriffs am höchsten erscheint.

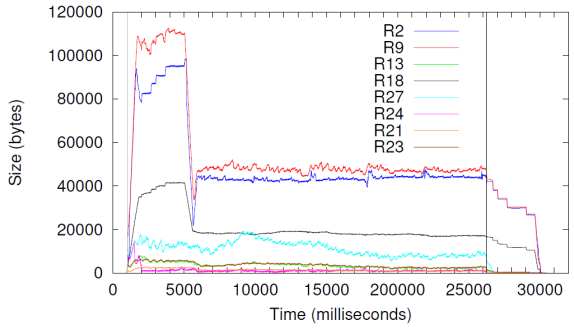

Abbildung 5.7: PIT-Größe bestimmter Router (originale Implementierung) [4]

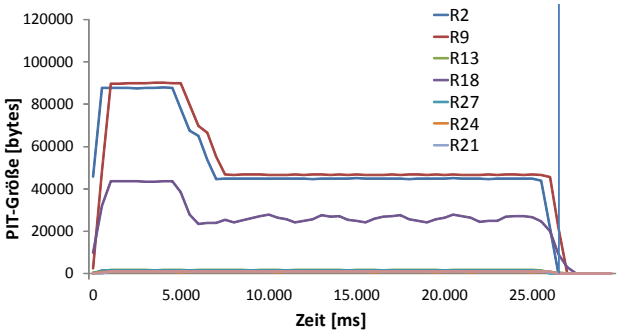

Abbildung 5.8: PIT-Größe bestimmter Router (eigene Implementierung)

Im direkten Vergleich ist ersichtlich, dass die eigene Implementierung (Abbildung 5.8) der originalen 5.7 sehr ähnelt. Die Router R2 (blau) und R9 (rot) haben in beiden Messreihen durch den simulierten Angriff einen starken Anstieg. In der eigenen Implementierung liegen die Maximal-Werte jedoch mit 90.000-95.000 *bytes* PIT-Einträgen etwas niedriger. Im betrachteten Zeitintervall weisen die PIT-Einträge der Router nach einsetzender Wirkung der Abwehrmethode einen ähnlichen Verlauf auf.Geringfügige Abweichungen, wie im Falle des Producer-Routers R27 (türkis), können auf das Routing-Verhalten des Simulators zurückzuführen sein. Dieser Effekt kann vernachlässigt werden, da es keinen Zusammenhang zur Wirkungsweise der Abwehrmethode gibt.

Da sich die Messwerte bzw. die Abbildungen 5.10 und 5.9 der Poseidon-Distributed-Methode kaum von der Local-Variante unterscheiden und die Router ähnliche Werte aufweisen, kann von einer gleichen Wirkungsweise der Abwehrmethode ausgegangen werden.

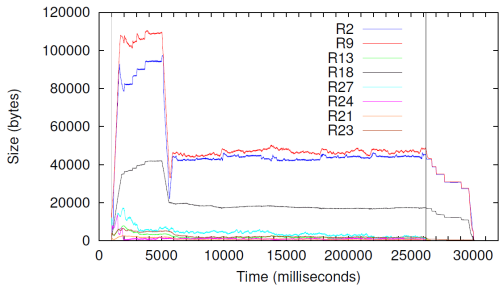

Abbildung 5.9: Poseidon Distributed PIT-Größe bestimmter Router (originale Implementierung) [4]

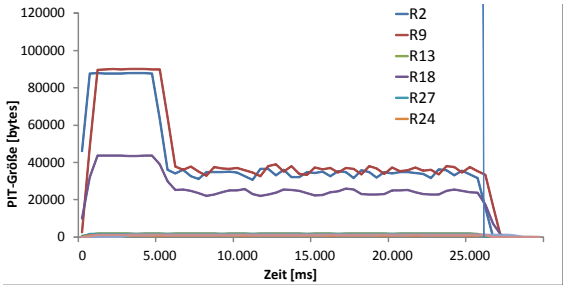

Abbildung 5.10: Poseidon Distributed PIT-Größe bestimmter Router (eigene Implementierung)

KAPITEL 6

Vergleichende Evaluierung und Bewertung

Nachdem die eigenen Implementierungen mit den jeweiligen Originalen auf Gleichheit geprüft wurden, sollen die Abwehrmethoden nun miteinander verglichen werden, um sie nach Effizienz und Wirkungsweise sortieren zu können. Um Nebeneffekte auszuschließen, werden für alle Tests gleiche Topologien und Szenarien verwendet.

Für die anschließende Bewertung werden bestimmte Leistungswerte als Messwerte zu Grunde gelegt. Diese werden im Folgenden näher erläutert:

6.1 Leistungswerte

PIT-Last

Da ein Interest-Flooding-Angriff versucht, die Pending Interest Table eines Routers zu füllen, ist die Anzahl der PIT-Einträge ein Indikator für den Zustand eines angegriffenen Routers. Daraus ergibt sich die Möglichkeit zu prüfen, ob eine Abwehrmethode in der Lage ist, die Anzahl der angreifenden PIT-Einträge während eines Angriffes gering zu halten, so dass diese einer normalen Netzwerkaktivität entsprechen.

Der Leistungswert wird während der Abwehr eines Angriffs auf jedem Router des NDN-Netzwerks in Form der absoluten Anzahl von PIT-Einträgen gemessen.

Interests-Daten-Verhältnis

Eine Flutung der PIT durch einen Angriff auf einen Router, kann zur Folge haben, dass Interests legitimer Consumer verworfen werden. Dadurch erhalten diese auf anfragende Interests weniger bzw. keine Daten aus dem NDN-Netzwerk. Demzufolge ist das Verhältnis von Interests zu Daten eines Consumers ein Indikator für eine erfolgreiche Datenzustellung trotz aktiven Angriffs.

Dieser Leistungswert wird für jeden legitimen Consumer außerhalb des NDN-Netzwerks als Verhältnis von Interests zu Daten gemessen. Außerdem wird geprüft, ob ein bestimmter Mindestwert erreicht wird, wodurch ein Consumer als „zufriedener Nutzer“ gelten kann.

Dieser Limit-Wert liegt bei etwa 90% des Verhältnisses von Interests zu Daten bei normaler Netzwerkaktivität.

Content Caching

Die Eigenschaft des Content Caching innerhalb eines NDN-Netzwerks ermöglicht es, dass Daten auf dem Pfad vom Producer zum Consumer von Routern zwischengespeichert werden. Während eines Angriffs können so Daten, die auf nicht unmittelbar vom Angriff betroffenen Routern gespeichert sind, weiterhin an anfragende Consumer verteilt werden. Es wird geprüft, ob diese Eigenschaft während der Abwehr eines Interest-Flooding-Angriffs Bestand hat.

Die Eigenschaft des Content Caching bleibt erhalten, wenn die Datenrate trotz Aktivierung der Abwehrmethode konstant bleibt. Dafür wird der Leistungswert als Anzahl der ankommenden Daten bei legitimen Consumern während eines Angriffs gemessen. Zuvor wird über eine gewisse Zeitdauer eine normale Netzwerkaktivität simuliert, wodurch Daten bereits zwischengespeichert werden.

6.2 Simulationsaufbau

6.2.1 Topologien

Die Topologien beschreiben Netzwerke, die durch ihren Aufbau und ihre Gliederung in Consumer, Router und Producer, Messungen der zuvor beschriebenen Leistungswerte für die jeweiligen Verfahren ermöglichen.

Abbildung 6.1 zeigt eine Topologie, die für die Messung der PIT-Last der Router genutzt wird. An drei Routern können verschiedene Szenarien getestet werden:

- alle benachbarten Consumer sind legitim,
- einer der drei Consumer ist ein Angreifer,
- alle benachbarten Consumer sind Angreifer.

In einem weiteren Testfall wird ein NDN-Netzwerk simuliert, bei dem die Router ringförmig angeordnet, miteinander und außerdem jeder dieser Router mit einem weiteren Consumer verbunden sind (Abbildung 6.2). Dieses Szenario wurde gewählt, da jeder Router unabhängig über einen angrenzenden Angreifer entscheiden kann.

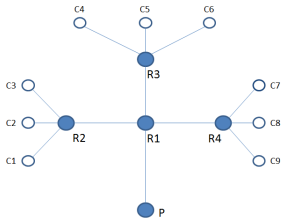

Abbildung 6.1: Topologie zur Messung der PIT-Last

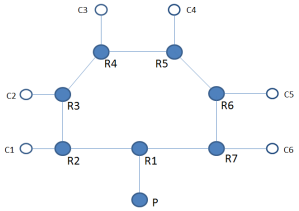

Abbildung 6.2: Topologie zur Messung der PIT-Last (Lokal)

In der Abbildung 6.3 wird ein Netzwerk aufgezeigt, das zur Messung des Interests-Daten-Verhältnisses aller Consumer genutzt wird. Dafür wird der einzige Router des NDN-Netzwerks von drei legitimen sowie von drei angreifenden Consumern begrenzt. Es wird das Verhältnis von Interests zu Daten der legitimen Consumer während eines Angriffs am selben Knotenpunkt gemessen.

Abbildung 6.4 zeigt eine weitere Topologie zur Messung des Verhältnisses von Interests zu Daten. Dabei sind alle vier Consumer an den Router R2 angeschlossen, der über den Router R1 mit dem Producer verbunden ist. Die Hälfte der Consumer sind dabei legitime Consumer, die andere Hälfte angreifende Knoten. Diese Topologie wird verwendet, da sie für Abwehrmethoden mit einem Pro-Interface-Verfahren aufgrund des „Interface-Flaschenhalses“ zwischen R2 und R1 eine Schwierigkeit darstellt.

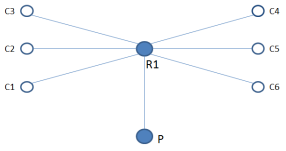

Abbildung 6.3: Topologie zur Messung des Interests-Daten-Verhältnisses 1

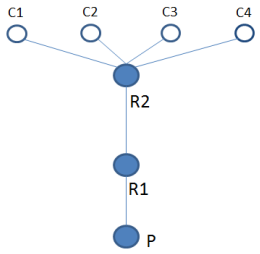

Abbildung 6.4: Topologie zur Messung des Interests-Daten-Verhältnisses 2

Zur Messung des Content-Caching-Leistungswerts wird ein NDN-Netzwerk angenommen, bei dem alle Router in einer Reihe aufsteigend miteinander verbunden sind (Abbildung 6.5).

Am Router R1 befinden sich drei angreifende und ein legitimer Consumer. Mit den beiden verbleibenden Routern ist jeweils ein legitimer Consumer verbunden. Durch diese Konstellation der Topologie kann über ein bestimmtes Zeitintervall eine normale Netzwerkaktivität simuliert werden bei der jeder Router die bereits angefragten Daten zwischenspeichert. Daraufhin wird der Knoten R1 von den drei angreifenden Consumern attackiert.

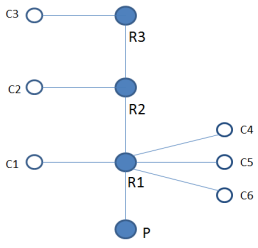

Abbildung 6.5: Topologie zur Messung von Content Caching

6.2.2 Szenario

Neben den genannten Topologien wird ein bestimmtes Szenario festgelegt, um die beschriebenen Leistungswerte verlässlich messen und als Basis für einen späteren Vergleich der Abwehrmethoden heranziehen zu können.

Die Master-Datei wird entsprechend den in der Topologie gegebenen Faktoren – Angreifer- und Consumer-Anzahl – geändert. Die verbleibenden Faktoren werden durch die Szenario-Datei wie folgt vorgegeben:

Zunächst werden die Applikationen der jeweiligen Knoten (Consumer, Router, Producer) mit einem Interest-LifeTime-Wert von einer Sekunde initialisiert. Der datenbringende Präfix des Producers wird auf "/good"gesetzt, da ausschließlich legitime Consumer Daten erhalten sollen. Die Paketgröße der Daten wird auf 1100 *bytes* festgelegt.

```
1   AppHelper evilAppHelper ("DdosDdosApp");
2   evilAppHelper.SetAttribute ("Evil", BooleanValue (true));
3   evilAppHelper.SetAttribute ("LifeTime", StringValue ("1s"));
4   evilAppHelper.SetAttribute ("DataBasedLimits", BooleanValue (true));
5   AppHelper goodAppHelper ("DdosDdosApp");
6   goodAppHelper.SetAttribute ("LifeTime",  StringValue ("1s"));
7   goodAppHelper.SetAttribute ("DataBasedLimits", BooleanValue (true));
8   AppHelper ph ("ns3::ndn::Producer");
9   ph.SetPrefix ("/good");
10  ph.SetAttribute ("PayloadSize", StringValue("1100"));
```

Quellcode 6.1: Szenario-Datei: Initialisierung

Anschließend werden die einzelnen Applikationen den Knoten zugewiesen und installiert, wobei jeder legitime Consumer den Präfix „/good“ und analog jeder Angreifer den Präfix „/evil“ erhält. Außerdem wird die Interest-Rate eines legitimen Consumers je nach Netzwerkaktivität berechnet, so dass Nebeneffekte vermieden werden. Die Rate der Angreifer wird konstant auf 0,001 Sekunden definiert. Abschließend werden die Producer-Knoten installiert und die jeweiligen Applikationen zeitgleich gestartet.

```
1  for (NodeContainer::Iterator node = goodNodes.Begin (); node != goodNodes.End ();
       node++)
2  {   ApplicationContainer goodApp;
3      goodAppHelper.SetPrefix ("/good/"+Names::FindName (*node));
4      goodAppHelper.SetAttribute ("AvgGap", TimeValue (Seconds (1.100 /
           maxNonCongestionShare)));
5      goodApp.Add (goodAppHelper.Install (*node));
6      UniformVariable rand (0, 1);
7      goodApp.Start (Seconds (0.0) + Time::FromDouble (rand.GetValue (), Time::S));
8  }
9  for (NodeContainer::Iterator node = evilNodes.Begin (); node != evilNodes.End ();
       node++)
10 {   ApplicationContainer evilApp;
11     evilAppHelper.SetPrefix ("/evil/"+Names::FindName (*node));
12     evilAppHelper.SetAttribute ("AvgGap", TimeValue (Seconds (0.001)));
13     evilApp.Add (evilAppHelper.Install (*node));
14     UniformVariable rand (0, 1);
15     evilApp.Start (Seconds (0.0)+Time::FromDouble (rand.GetValue (),Time::MS));
16     evilApp.Stop  (Seconds (900.0)+Time::FromDouble (rand.GetValue (),Time::MS));
17 }
18 ph.Install (producerNodes);
```

Quellcode 6.2: Szenario-Datei: Installation

6.3 Ergebnisse

Nachdem der Simulationsaufbau nun eindeutig definiert wurde, wurden alle Abwehrmethoden nacheinander für alle Topologien simuliert.

Im Folgenden werden nun bei allen Abwehrmethoden die ermittelten Messwerte bzw. Ergebnisse bezüglich

- PIT-Last
- Interests-Daten-Verhältnis
- Content Caching

beschrieben.

6.3.1 DDoS-Abwehrmethode

PIT-Last

In den Abbildungen 6.6 und 6.7 ist der Verlauf der Anzahl der PIT-Einträge über ein Zeitintervall von 30 Sekunden für alle Router der jeweiligen Topologie PIT-Last und PIT-Last (Lokal) dargestellt.

Die Abbildung 6.6 zeigt die Messwerte der PIT-Last-Topologie, die sich aus drei verschiedenen Consumer/Angreifer-Konstellationen ergeben haben: Router4 ist mit drei Angreifern anfangs einer sehr hohen PIT-Belastung ausgesetzt. Router3 – mit einem Angreifer und zwei Consumern – und Router2 – mit drei Consumern – starten bei einem Wert von rund 50 und 20 PIT-Einträgen. Router1 verbindet alle diese Router mit dem Producer, wodurch dieser anfangs die höchste Belastung erfährt. Durch die Limitierung der Präfixe der Angreifer auf dem Producer-Knoten bis zum jeweiligen Router fällt die Anzahl der PIT-Einträge des Router4 in der folgenden Sekunde auf einen Wert von 0, wodurch keine weiteren Interests der drei Angreifer weitergeleitet werden. Der Wert des Router1 fällt ähnlich ab und verbleibt – analog zu den beiden anderen Routern – auf einem Wert von 20, was einer normalen Netzwerkaktivität entspricht.

Die Abbildung 6.7 stellt einen ähnlichen Graphen für die Topologie PIT-Last (Lokal) dar, bei der die Angreifer mit den Routern 3, 5 und 7 verbunden sind. Diese Router weisen anfangs einen hohen PIT-Wert auf. Analog zu der vorhergehenden Abbildung fallen die Werte dieser Router innerhalb der folgenden Sekunde aufgrund der Limitierung der angreifenden Präfixe auf den Routern auf einen Wert von 0. Der Producer-Router, Router1, und der Verbindungsknoten, Router6, weisen anfangs durch das Weiterleitungsverhalten einen ähnlichen hohen Wert auf, der durch die Limitierung der Angreifer jedoch auf einen Wert von 20 fällt.

Die Verläufe der PIT-Belastungen der einzelnen Router in den beiden Topologien zeigen, dass die DDoS-Abwehrmethode einen simulierten Interest-Flooding-Angriff durch Senkung der angreifenden PIT-Einträge abwehren kann.

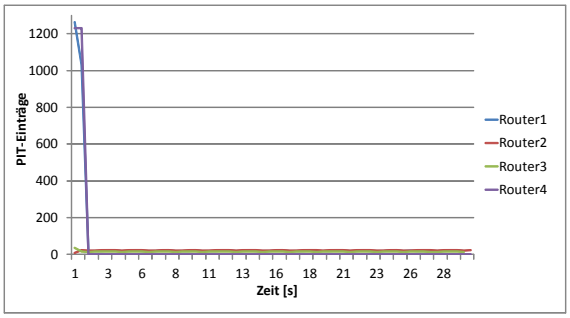

Abbildung 6.6: PIT-Last bei der DDoS-Methode

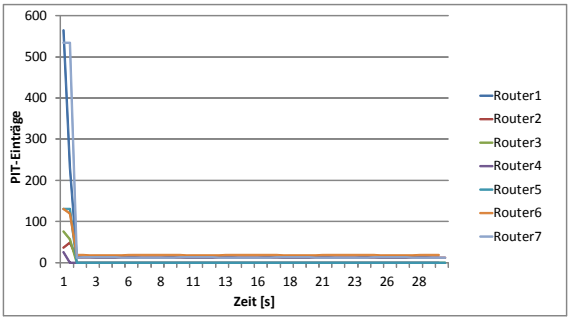

Abbildung 6.7: PIT-Last (Lokal) bei der DDoS-Abwehrmethode

Interests-Daten-Verhältnis

In Abbildung 6.8 wird das prozentuale Verhältnis von Interests zu Daten von drei legitimen Consumern über ein Zeitintervall von 30 Sekunden während eines Angriffs dargestellt. Es fällt auf, dass das Verhältnis nach einer Sekunde bereits bei über 90% liegt. Das Verhältnis steigt in den folgenden Sekunden weiter an und strebt gegen einen Wert von 100%. Es kann davon ausgegangen werden, dass die Limitierung der drei angreifenden Präfixe durch die Abwehrmethode diesen Wert ermöglicht.

Die Abbildung 6.9 zeigt einen ähnlichen Graphen, wobei bei dieser Topologie zwei von vier am Router2 angrenzenden Consumer nicht legitim sind. Analog zum vorherigen Messergebnis kann davon ausgegangen werden, dass nur durch die Limitierung der Angreifer-Präfixe ein solcher Anstieg erreicht werden konnte.

Die Verwendung der DDoS-Abwehrmethode ermöglicht für legitime Consumer in beiden Topologien ein Verhältnis von Interests zu Daten von bis zu 100%.

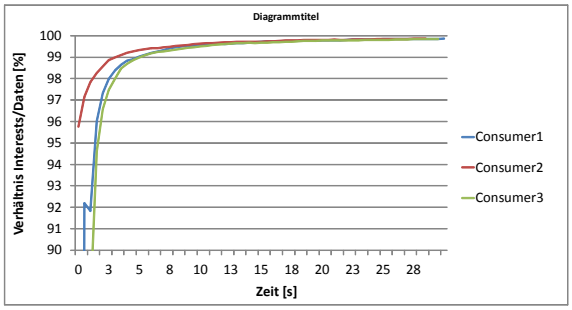

Abbildung 6.8: Interests-Daten-Verhältnis 1 bei der DDoS-Abwehrmethode

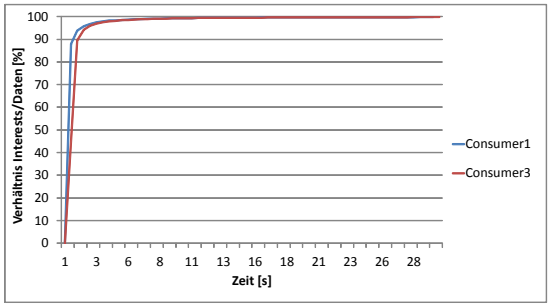

Abbildung 6.9: Interests-Daten-Verhältnis 2 bei der DDoS-Abwehrmethode

Content Caching

In der Abbildung 6.10 wird die Fähigkeit des Content Caching während eines Angriffes und dessen Abwehr mit Hilfe der Anzahl der Daten über ein Zeitintervall von 30 Sekunden

dargestellt, wobei die ersten zehn Sekunden für die Simulation einer normalen Netzwerkaktivität verwendet werden. Es fällt auf, dass alle Werte bei einem Wert von rund 148 Daten pro Sekunde starten. Dieser Wert schwankt über die folgenden Sekunden leicht, bleibt jedoch insgesamt konstant. Das heißt, die Eigenschaft des Content Caching bleibt während der Abwehr des Angriffes durch die DDoS-Methode erhalten.

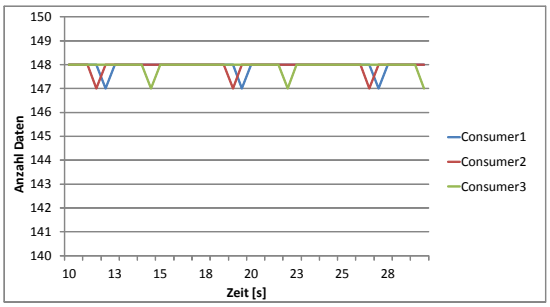

Abbildung 6.10: Content Caching bei der DDoS-Abwehrmethode

6.3.2 Interest-Traceback

Für diese und alle nachfolgenden Abwehrmethoden gelten durch die verwendeteten Topologien und Diagramme die gleichen Rahmenbedingungen, so dass diese im Folgenden nicht erneut erläutert werden.

PIT-Last

In der Abbildung 6.11 wird der Graph für die PIT-Belastung für die Interest-Traceback dargestellt. Alle Knoten mit angrenzenden Angreifern weisen eine hohe PIT-Belastung auf, wobei die Höhe des Anfangswerts durch die Angreiferzahl beeinflusst wird. Durch die Limitierung aller Interfaces mit kurz vor dem Timeout liegenden Interests innerhalb der PIT fallen die Werte der Router 1 und 2 auf einen Wert, der einer normalen Netzwerkaktivität entspricht. Die Werte von Router3 sinken jedoch auf 0, wodurch keinerlei Interests weitergeleitet werden, obwohl diese auch von legitimen Consumern stammen.

Die Abbildung 6.12 zeigt zunächst einen ähnlichen Verlauf für die Lokal-Topologie. Allerdings sinken durch die Limitierung alle Router-Werte auf 0, so dass ein Großteil des Netzwerkverkehrs zusammenbricht.

Die Messwerte zeigen, dass die Interest-Traceback-Methode in der Lage ist, einen Angriff mit einhergehender Steigerung der PIT-Einträge abzuwehren. Es fällt jedoch auf, dass die PIT-Last dabei unter den Wert einer normalen Netzwerkaktivität fällt.

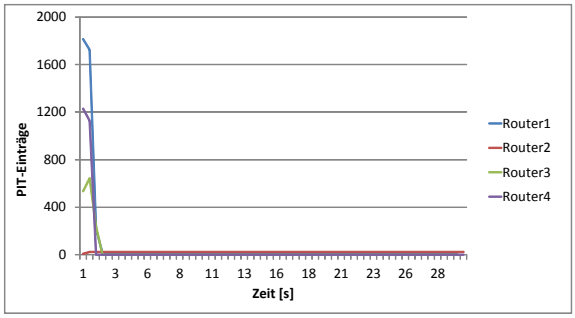

Abbildung 6.11: PIT-Last bei der Interest-Traceback-Methode

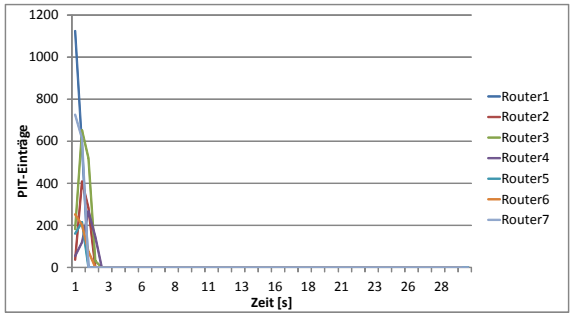

Abbildung 6.12: PIT-Last (Lokal) bei der Interest-Traceback-Methode

Interests-Daten-Verhältnis

Abbildung 6.13 stellt das Verhältnis von Interests zu Daten für alle legitimen Consumer dar.

Alle Werte beginnen bei 0, da es eine gewisse Zeit dauert, bis auf Interests Daten folgen. Nach einem vorübergehenden Anstieg sinken die Werte durch den zuvor beschriebenen Zusammenbruch des Netzwerkverkehrs mit einsetzender Abwehr schrittweise auf 0.

Die Abbildung 6.14 zeigt einen ähnlichen Graphenverlauf für die Topologie Interests-Daten-Verhältnis 2.

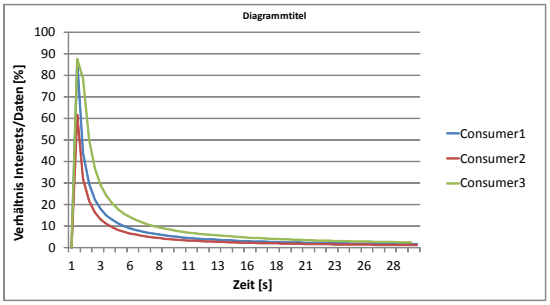

Abbildung 6.13: Interests-Daten-Verhältnis 1 bei der Interest-Traceback-Methode

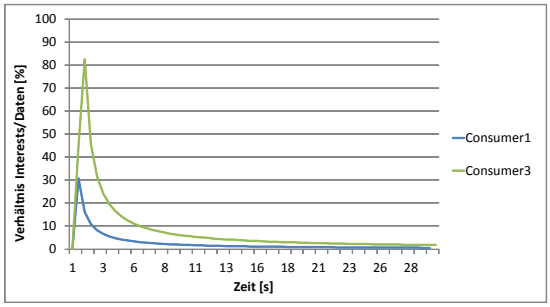

Abbildung 6.14: Interests-Daten-Verhältnis 2 bei der Interest-Traceback-Methode

Content Caching

In der Abbildung 6.15 wird geprüft, ob die Abwehrmethode die Eigenschaft des NDN-Prinzips des Content-Caching unterstützt, wobei in den ersten 10 Sekunden eine normale Netzwerkaktivität simuliert wird. Beginnend bei einer Datenrate von 148 pro Sekunde fallen die Werte in den folgenden drei Sekunden auf einen Wert von 0, verursacht durch den zuvor beschriebenen Zusammenbruch. Somit ist die Eigenschaft Content-Caching nicht mehr gegeben.

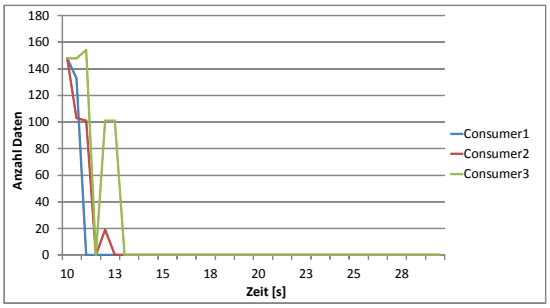

Abbildung 6.15: Content Caching bei der Interest-Traceback-Methode

6.3.3 Threshold-based detecting and mitigating

PIT-Last

In der Abbildung 6.16 wird der Graph zur PIT-Last für jeden Router der Topologie dargestellt, wobei diese versuchen, den Interest-Flooding-Angriff durch die TDM-Methode abzuwehren.

Nach einem zwischenzeitlichen Anstieg der PIT-Last durch den Angriff, fallen die Werte der betroffenen Router nach Drosselung der angreifenden Präfixe durch bestimmte Limits in den folgenden Sekunden auf einen Normalwert von rund 20. Router4, mit drei anliegenden Angreifern, wird total und somit bis zu einem Wert von 0 limitiert.

Die Abbildung 6.17 zeigt einen ähnlichen Graphen-Verlauf für die PIT-Belastung innerhalb der Lokal-Topologie. Dabei fallen die Werte aller Router auf einen für eine normale Netzwerkaktivität typischen Wert von 20. Nur Router5 hingegen, der kein Verbindungsknoten zwischen einem legitimen Consumer und dem Producer ist, sinkt auf einen Wert von 0.

Die beiden Verläufe der PIT-Belastungen für jeden Router der jeweiligen Topologien zeigen, dass die TDM-Methode einen simulierten Interest-Flooding-Angriff abwehren kann.

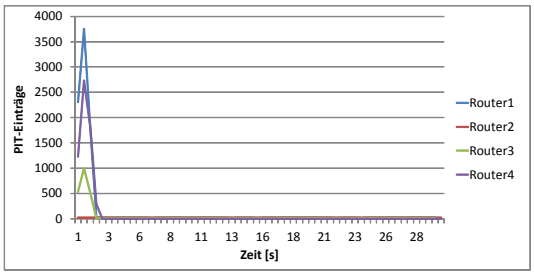

Abbildung 6.16: PIT-Last bei der TDM-Methode

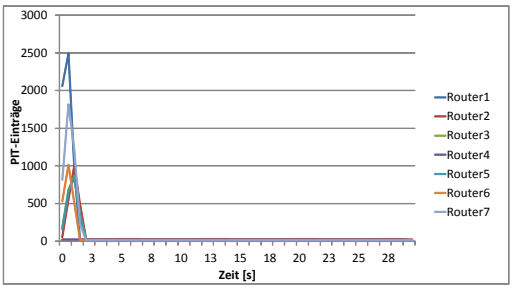

Abbildung 6.17: PIT-Last (Lokal) bei der TDM-Methode

Interests-Daten-Verhältnis

In der Abbildung 6.18 wird der Verlauf des Interests-Daten-Verhältnisses für ein Zeitintervall dargestellt. Die Verhältniswerte steigen in den ersten Sekunden durch die Drosselung der angreifenden Präfixe und die somit für legitime Interests freie PIT auf einen Wert von rund 100%.

Die Abbildung 6.19 zeigt für die Topologie „Interests-Daten-Verhältnis 2“ ein ähnliches Bild des Verhältnisverlaufs.

Beide Diagramme zeigen, dass durch bei der Abwehr des Angriffs mittels TDM-Verfahren das Interests-Daten-Verhältnis aller legitimen Consumer auf einem hohen Wert von mehr als 99% liegt.

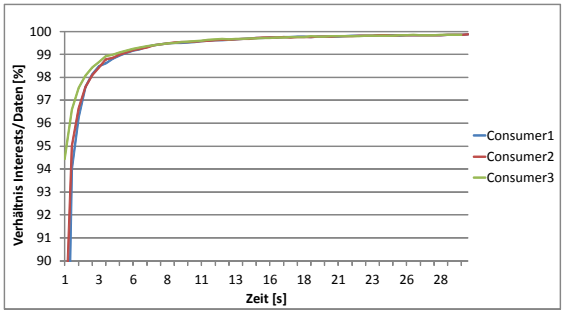

Abbildung 6.18: Interests-Daten-Verhältnis 1 bei der TDM-Methode

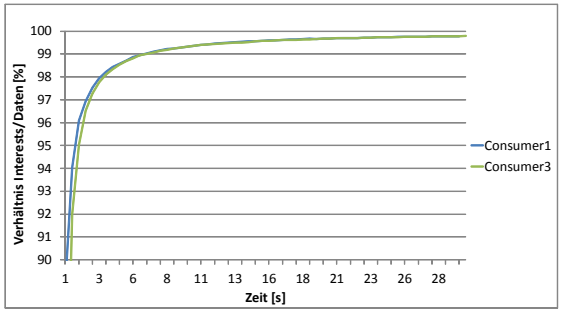

Abbildung 6.19: Interests-Daten-Verhältnis 2 bei der TDM-Methode

Content Caching

In Abbildung 6.20 wird der Graph zur Prüfung der Content-Caching-Fähigkeit dargestellt. Die Werte aller drei legitimen Consumer beginnen bei rund 148 Daten pro Sekunde und bleiben trotz gleichzeitigem Angriff bei diesem Wert, was auf Erhalt des Content-Caching schließen lässt.

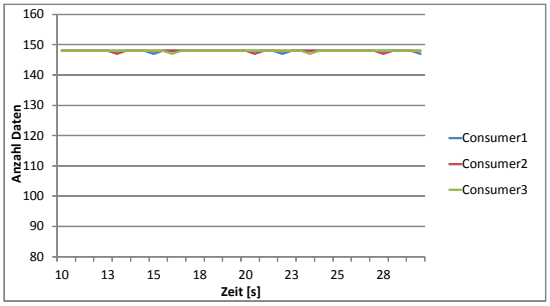

Abbildung 6.20: Content Caching bei der TDM-Methode

6.3.4 Resource Allocation

PIT-Last

In der Abbildung 6.21 wird der Graph zum Verlauf der PIT-Last bei der Verwendung der Resource-Allocation-Methode dargestellt.

Da bei dieser Abwehrmethode eine Obergrenze für die PIT-Größe von 1200 gesetzt wird, kann der Graph nur bis zu diesem Wert steigen. Je nach Angreiferzahl steigen die Werte der jeweiligen Router zunächst auf einen entsprechenden Wert und verbleiben dort bis zum Simulationsende.

Die Abbildung 6.22 zeigt einen ähnlichen PIT-Verlauf aller Router innerhalb der Lokal-Topologie. Neben dem Producer-Router, Router1, und den Routern mit angrenzenden Angreifer – Router3, Router5 und Router7 – haben die Verbindungsknoten, Router2 und Router6, aufgrund der weiterleitenden Funktion eine erhöhte PIT-Last. Router4 hingegen hat durch seine spezielle Lage innerhalb des Netzwerks keine weiterleitende Funktion, so dass nur die Aktivität eines legitimen Consumers von rund 20-70 PIT-Einträgen gemessen wird.

Die Verläufe der einzelnen Graphen der Router zeigen, dass die Abwehrmethode Resource Allocation einen Angriff durch Senkung der angreifenden PIT-Einträge nicht abwehren kann.

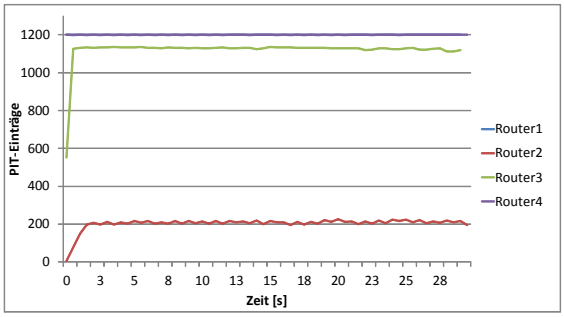

Abbildung 6.21: PIT-Last bei der Resource-Allocation-Methode

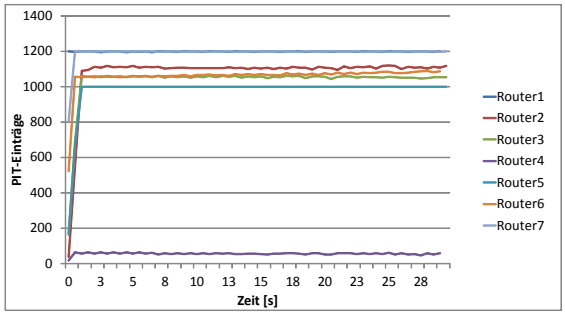

Abbildung 6.22: PIT-Last (Lokal) bei der Resource-Allocation-Methode

Interests-Daten-Verhältnis

In der Abbildung 6.23 wird der Verlauf des prozentualen Verhältnisses von Interests zu Daten aller legitimen Consumer dargelegt. Im vorgegebenen Zeitintervall steigen die Werte bis ca. 100%.

Die Abbildung 6.24 zeigt einen anderen Verlauf des Verhältnisses für die Topologie „Interests-Daten-Verhältnis 2“. Das Verhältnis beider legitimer Consumer beginnt bei einem Wert von 0% und steigt innerhalb der ersten zwei Sekunden auf einen Wert von mehr als 50%. Durch die gleichberechtigte Weiterleitung aller Interests durch die Abwehrmethode am Router2 steigt das Verhältnis jedoch nicht weiter an, sondern bleibt bei einem ähnlichen Wert von 50-60%.

Die Messwerte zeigen, dass das Verhältnis von Daten zu Interests der legitimen Consumer aufgrund topologischer Gegebenheiten sinken kann.

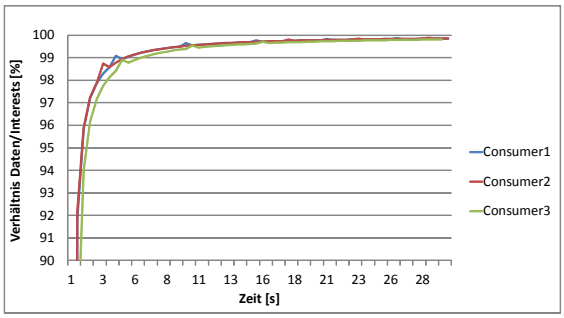

Abbildung 6.23: Interests-Daten-Verhältnis 1 bei der Resource-Allocation-Methode

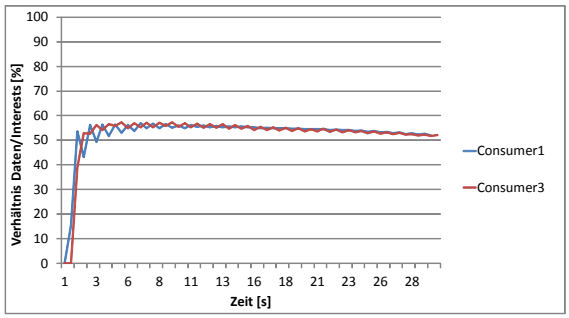

Abbildung 6.24: Interests-Daten-Verhältnis 2 bei der Resource-Allocation-Methode

Content Caching

In der Abbildung 6.25 wird geprüft, ob die Eigenschaft des Content Caching während der Abwehr des Angriffs bewahrt bleibt.

Mit dem Einsetzen des Angriffs fallen die Werte aller Consumer auf rund 15 und schwanken innerhalb des verbleibenden Zeitintervalls zwischen zwei Werten, wobei diese insgesamt sinken.

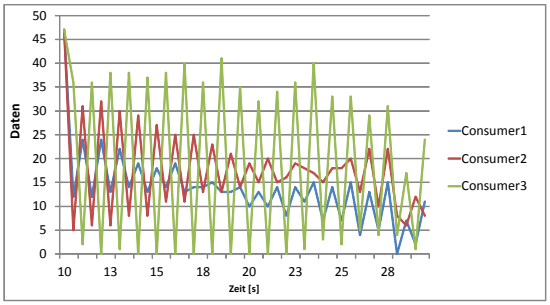

Abbildung 6.25: Content Caching bei der Resource-Allocation-Methode

6.3.5 Poseidon Local

PIT-Last

In der Abbildung 6.26 wird der Verlauf der PIT-Last jedes Routers für die Poseidon-Local-Methode dargestellt.

Die Werte des Producer-Routers1 und des Routers4 mit drei anliegenden Angreifer sinken durch Limitierung der angreifenden Interfaces auf den jeweiligen Routern. Router3 mit nur einem angrenzenden Angreifer steigt in der selben Zeit auf einen Wert von 300. Router2 hält während der gesamten Simulation einen Wert von 20.

Die Abbildung 6.27 für PIT-Last (Lokal) zeigt einen ähnlichen Verlauf. Die Router mit einem benachbarten Angreifer fallen durch die Limitierung des Interfaces in den ersten Sekunden auf einen Wert von 0-20. Router, die ausschließlich eine weiterleitende Funktion besitzen, steigen auf einen Wert von 300. Diese Werte bleiben bis zum Ende der Simulation konstant.

Die PIT-Belastungen beider Topologien zeigen, dass die Abwehrmethode einen Angriff in Form von Steigerung der PIT-Einträge mehrheitlich abwehren kann. Es fällt jedoch auf, dass vereinzelte Router durch ihre Lage innerhalb der Topologie nicht auf den Wert einer normalen Netzwerkaktivität gedrosselt werden und somit weiterhin eine erhöhte PIT-Belastung aufweisen.

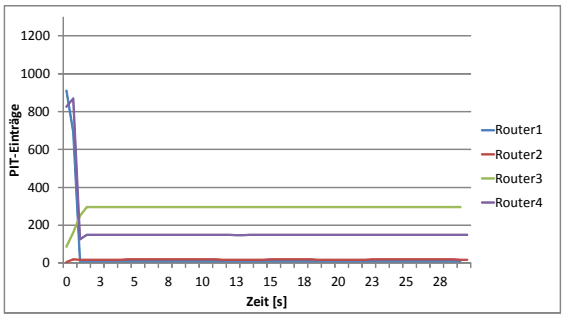

Abbildung 6.26: PIT-Last bei der Poseidon-Local-Methode

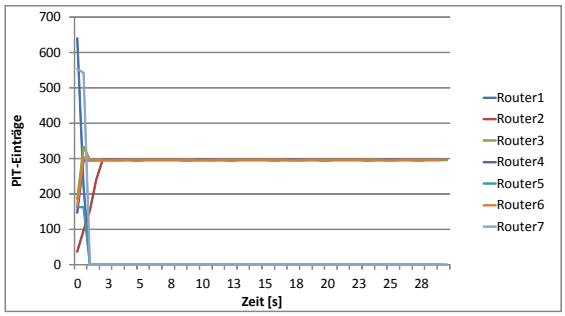

Abbildung 6.27: PIT-Last (Lokal) bei der Poseidon-Local-Methode

Interests-Daten-Verhältnis

In der Abbildung 6.28 wird der prozentuale Verlauf des Interests-Daten-Verhältnisses aller Consumer dargestellt.

Durch die Limitierung der drei Interfaces mit adjazenten Angreifern steigen die Werte auf rund 100%.

Die Abbildung 6.29 zeigt, dass die Werte beider legitimen Consumer zu jedem Zeitpunkt der Simulation bei 0 liegen. Dieses Verhalten ist auf den negativen Effekt des Interface-basierten Abwehrprinzips zurückzuführen. Da in diesem Fall die Weiterleitungswahrscheinlichkeit des

Interfaces von Router2 (an Router1) 0 ist, werden alle dahinter liegenden Consumer blockiert.

Aus den Messwerten wird deutlich ersichtlich, dass das Interests-Daten-Verhältnis stark von der Topologie abhängig ist und je nach Konstellation bis zu einem völligen Abbruch der Datenverteilung führen kann.

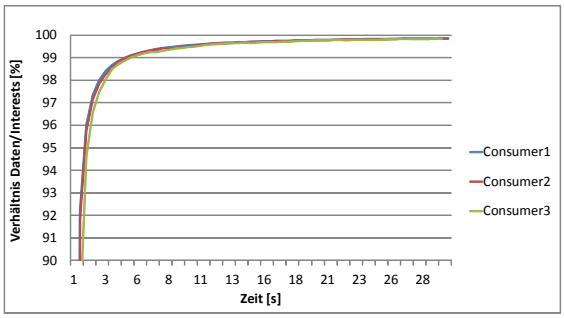

Abbildung 6.28: Interests-Daten-Verhältnis 1 bei der Poseidon-Local-Methode

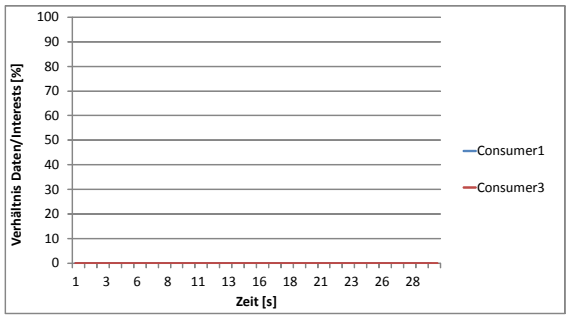

Abbildung 6.29: Interests-Daten-Verhältnis 2 bei der Poseidon-Local-Methode

Content Caching

In der Abbildung 6.30 wird die Prüfung des Erhalts der NDN-Eigenschaft des Content-Caching dargestellt. Nach einer zehnsekündigen Simulation bei normaler Netzwerkaktivität, wobei eine Datenrate von 148 existiert, wird die Abwehrmethode aktiviert. Da diese bis auf leichte Schwankungen diese Datenrate während des gesamten Vorgangs halten kann, kann davon ausgegangen werden, dass die Eigenschaft erhalten bleibt.

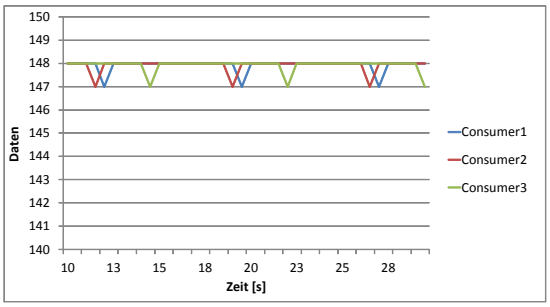

Abbildung 6.30: Content Caching bei der Poseidon-Local-Methode

6.3.6 Poseidon Distributed

PIT-Last

In der Abbildung 6.31 wird der Verlauf der PIT-Last jedes Routers der Topologie PIT-Last für die Poseidon-Distributed-Methode dargestellt.

Die Werte des Producer-Routers1 und des Routers4 mit drei anliegenden Angreifer sinken durch Limitierung der angreifenden Interfaces auf den jeweiligen Routern. Router3 mit nur einem angrenzenden Angreifer steigt in der selben Zeit auf einen Wert von 150. Router2 hält während der gesamten Simulation einen Wert von 20.

Die Abbildung 6.32 zeigt einen ähnlichen PIT-Verlauf aller Router innerhalb der Lokal-Topologie.

Der Producer-Router, Router1, Router4 und die Router mit angrenzenden Angreifern – Router3, Router5 und Router7 – haben eine erhöhte PIT-Last. Während Router3 und Router4 nach kurzer Zeit einen Wert von 300 erreichen und dort verbleiben, sinken die Werte der anderen Router nach kurzer Zeit gegen 0.

Die beiden Verläufe zeigen die mehrheitliche Abwehr des Angriffs durch Senkungen der PIT-Einträge. Jedoch kommt es je nach topologischer Lage eines Routers zu einer erhöhten PIT-Last.

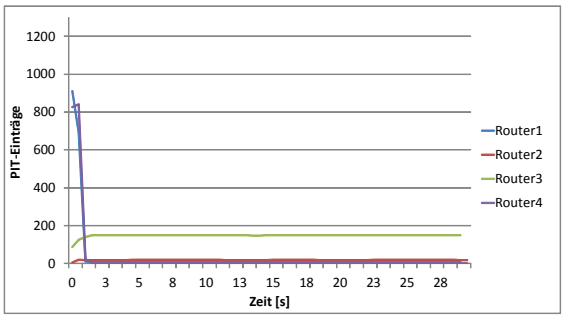

Abbildung 6.31: PIT-Last bei der Poseidon-Distributed-Methode

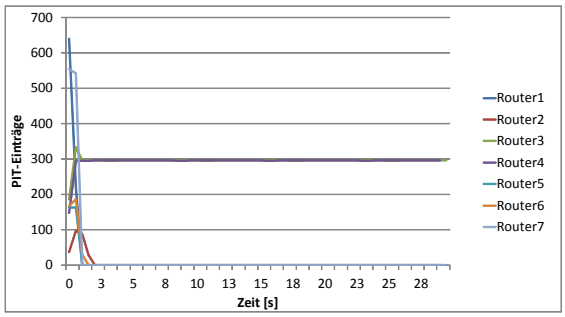

Abbildung 6.32: PIT-Last (Lokal) bei der Poseidon-Distributed-Methode

Interests-Daten-Verhältnis

In der Abbildung 6.33 wird der Verlauf des Interests-Daten-Verhältnisses der Consumer dargestellt.

Durch die Limitierung der Angreifer und die somit freie Kapazität der PIT steigen die Werte aller Consumer bis zum Ende der Simulation gegen 100%. Unter den Bedingungen der zweiten Topologie bleiben die Werte durch Blockierung des Interfaces von Router2 sehr niedrig.

Die Verläufe beider Messreihen zeigen, dass das Interests-Daten-Verhältnis der Consumer stark von der Topologie und der damit einhergehenden Konstellation der Router abhängt.

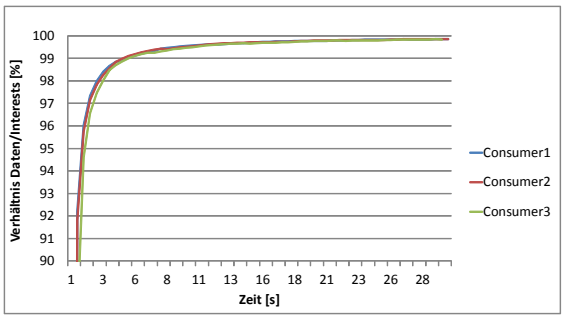

Abbildung 6.33: Interests-Daten-Verhältnis 1 bei der Poseidon-Distributed-Methode

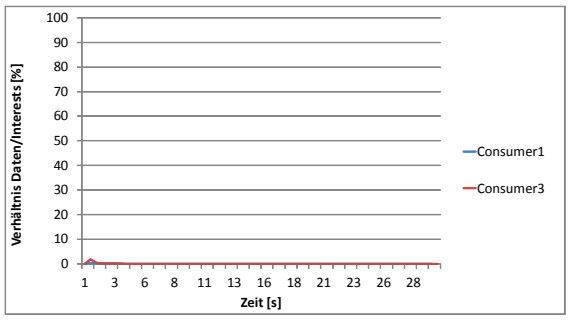

Abbildung 6.34: Interests-Daten-Verhältnis 2 bei der Poseidon-Distributed-Methode

Content Caching

Da die Abbildung 6.35 einen sehr ähnlichen Verlauf der Content-Caching-Eigenschaft wie die Abbildung 6.30 (Poseidon Local) zeigt, wird von einem gleichen Verhalten und dem Erhalt der Eigenschaft ausgegangen.

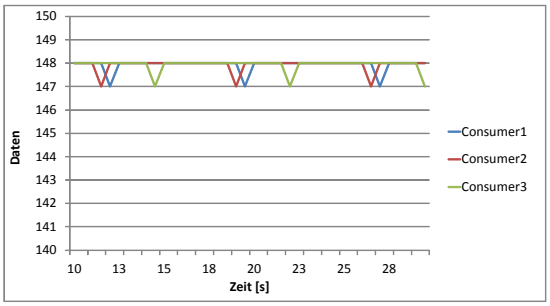

Abbildung 6.35: Content Caching bei der Poseidon-Distributed-Methode

6.3.7 Token Bucket with per-interface fairness

PIT-Last

In der Abbildung 6.36 werden die Graphen zur PIT-Last für die Token-Bucket-Methode während eines Angriffs für die jeweiligen Router dargestellt. Je nach Anzahl der angrenzenden Angreifer steigen die Werte der PIT-Last in den ersten Sekunden um 50 - 200 PIT-Einträge und fallen anschließend auf den Ausgangswert zurück. Da die Interests aller Interfaces nacheinander gleichberechtigt weitergeleitet werden, wiederholen sich die Werte je nach Verhältnis von legitimen Consumer zu Angreifer.

Die Abbildung 6.37 zeigt ein ähnlichen Verlauf der PIT-Belastung jedes Routers.

Die Messwerte zeigen, dass durch das Token-Bucket-Verfahren ein simulierter Interest-

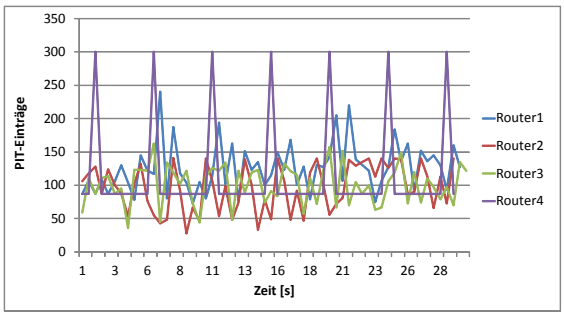

Abbildung 6.36: PIT-Last bei der Token-Bucket-Methode

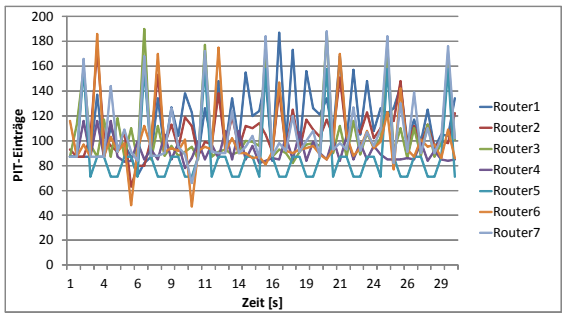

Abbildung 6.37: PIT-Last (Lokal) bei der Token-Bucket-Methode

Flooding-Angriff durch Senkung der PIT-Einträge nicht abgewehrt werden kann.

Interest-Daten-Verhältnis

In der Abbildung 6.38 wird der Verlauf der Verhältniswerte der legitimen Consumer dargestellt. In den ersten Sekunden steigen die Werte minimal auf 8-12% und fallen bis zum Ende der Simulation durch die gleichberechtigte Weiterleitung aller Interests, wobei die Anzahl angreifender Interests deutlich höher liegt.

Die Abbildung 6.39 zeigt, dass durch die Vielzahl angreifender Interests die Werte beider Consumer im Zeitintervall lediglich einen Wert von ca. 20% erreichen konnten.

Die beiden Verläufe der Token-Bucket-Methode weisen im Interests-Daten-Verhältnis einen prozentual niedrigen Wert auf, der keiner normalen Netzwerkaktivität entspricht.

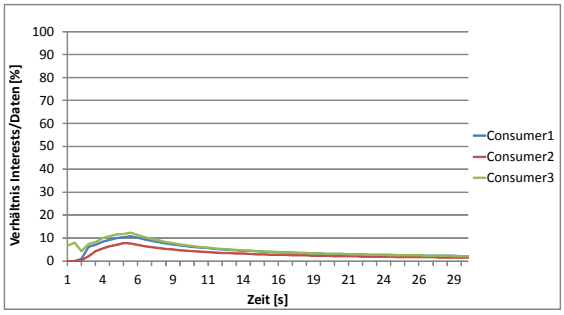

Abbildung 6.38: Interests-Daten-Verhältnis 1 bei der Token-Bucket-Methode

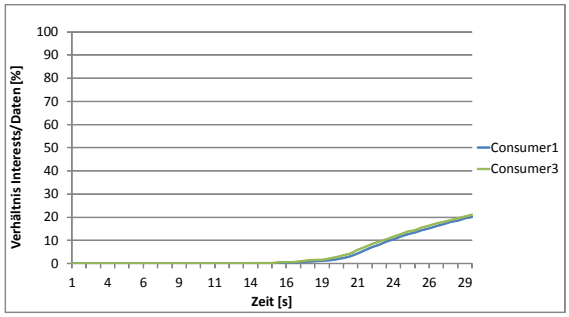

Abbildung 6.39: Interests-Daten-Verhältnis 2 bei der Token-Bucket-Methode

Content Caching

In der Abbildung 6.40 wird die Content-Caching-Eigenschaft geprüft.

Bei Einsetzen des Angriffs fallen die Werte aller Consumer auf nahezu 0 Daten und erreichen bis zum Ende des Zeitintervalls aufgrund der gleichberechtigten Weiterleitung nicht den

Wert der normalen Netzwerkaktivität. D.h. die Eigenschaft des Content Caching ist während der Abwehr durch die Token-Bucket-Methode nicht mehr gegeben.

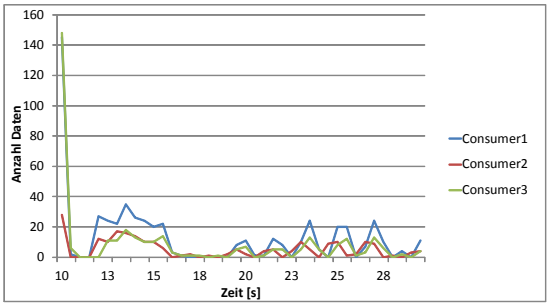

Abbildung 6.40: Content Caching bei der Token-Bucket-Methode

6.3.8 Satisfaction-based Interest acceptance

PIT-Last

In der Abbildung 6.41 wird der Verlauf der PIT-Belastung für die Interest-Acceptance-Methode dargestellt. Die PIT-Werte der einzelnen Router steigen bzw. fallen in einem scheinbar festen zeitlichen Rhythmus, wobei die Werte innerhalb eines verträglichen PIT-Belastungsrahmens bleiben. Die rhythmische Anstiegswiederholung basiert auf der durch die berechneten Wahrscheinlichkeitswerte erfolgenden Weiterleitung für die Interfaces. Der Anstieg ist dabei Folge eines zuvor stattgefundenen Timeouts.

Die Abbildung 6.42 zeigt einen sehr ähnlichen Verlauf für die PIT-Last (Lokal).

Die Messwerte zeigen, dass eine Abwehr durch die Interest-Acceptance-Methode stattfindet, wobei die PIT-Einträge durch Schwankungen leicht oberhalb normaler Netzwerkaktivität liegen können.

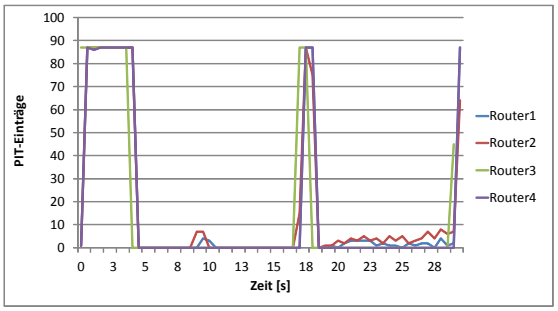

Abbildung 6.41: PIT-Last bei der Interest-Acceptance-Methode

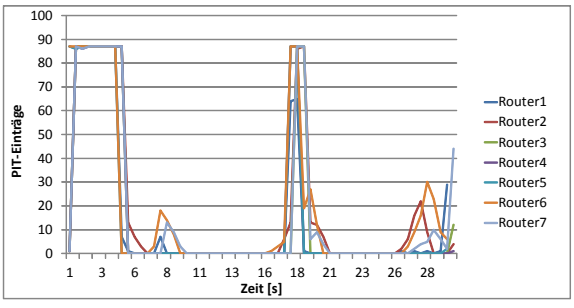

Abbildung 6.42: PIT-Last (Lokal) bei der Interest-Acceptance-Methode

Interests-Daten-Verhältnis

In der Abbildung 6.43 werden die Werte des Interests-Daten-Verhältnisses dargestellt, wobei sich der zuvor beschriebene zeitliche Rhythmus der PIT-Last darauf auszuwirken scheint.

Die Abbildung 6.44 zeigt, dass die Werte während der gesamten Simulation unter 0,2% liegen. Der Grund dafür ist die Wahrscheinlichkeitsberechnung des Knotens, Router2. Router1 erhält über verschiedene Interfaces Interests von zwei legitimen sowie zwei angreifenden Consumern. Dadurch wird für jedes Interface eine einzelne Wahrscheinlichkeit berechnet. Da die Interests jedoch gemeinsam über ein Interface zu Router2 gelangen, wird ausschließlich für dieses Interface am Router2 eine Wahrscheinlichkeit berechnet. Aufgrund der enormen Anzahl an angreifenden Interests, fällt der Wert sehr gering aus, wodurch wenige bis keine Interests an den Producer weitergeleitet werden.

Die Messwerte zeigen, dass das Verhältnis von Interests zu Daten je nach topologischen Gegebenheiten zwischen Werten von 0,2% bis 23% schwankt und somit stark unterhalb normaler Netzwerkaktivität liegt.

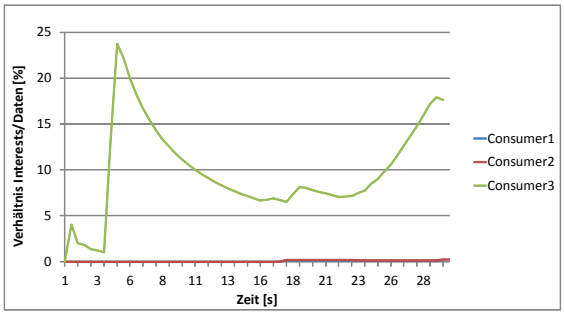

Abbildung 6.43: Interests-Daten-Verhältnis 1 bei der Interest-Acceptance-Methode

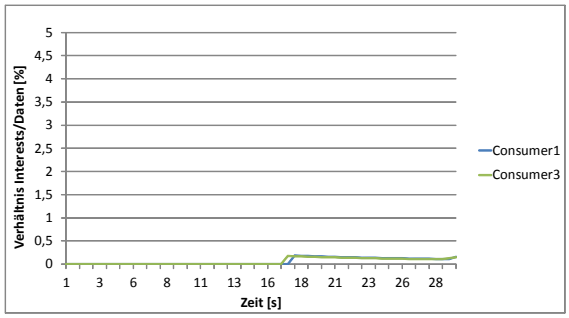

Abbildung 6.44: Interests-Daten-Verhältnis 2 bei der Interest-Acceptance-Methode

Content Caching

In der Abbildung 6.45 wird anhand der Datenanzahl gezeigt, ob die Eigenschaft des Content-Caching durch das NDN-Prinzip während der Abwehr gegeben bleibt. Die Werte fallen bei Start des Angriffs auf einen Wert um 0%. Nach einem vorübergehenden Anstieg folgt ein Zusammenbruch der Datenrate kurz vor dem Ende der Simulation, auf den ein erneuter Anstieg auf die davor erreichten Werte folgt.

Die Eigenschaft des Content Caching ist durch einen kompletten Abbruch bei Abwehr des Angriffs nicht mehr gegeben.

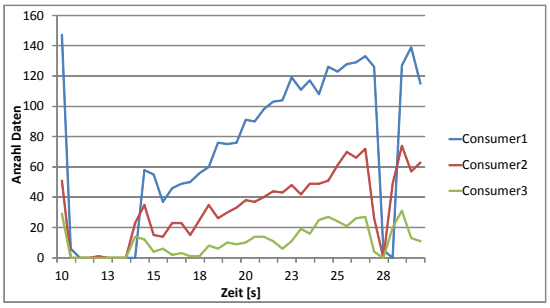

Abbildung 6.45: Content Caching bei der Interest-Acceptance-Methode

6.3.9 Satisfaction-based pushback

PIT-Last

In der Abbildung 6.46 wird der Verlauf der PIT-Belastung der einzelnen Router während der Abwehr durch die Interest-Pushback-Methode dargestellt. Alle Router des Netzwerks steigen anfangs von 100 PIT-Einträgen leicht an. In den folgenden Sekunden fallen die Werte aller Router als Effekt des vollständig verteilten Pushback-Mechanismus und der damit verbundenen Limitierung der Interfaces auf einen Wert von 0. Nach einem konstanten Verlauf bei diesem Wert kommt es zum Ende der Simulation zu einer leichten Erhöhung, die jedoch im Bereich einer normalen Netzwerkaktivität liegt.

Die Abbildung 6.47 zeigt einen ähnlichen Verlauf der PIT-Einträge.

Die Messwerte ergeben, dass die Interest-Pushback-Methode den simulierten Angriff mit Steigerung der angreifenden PIT-Einträge abwehren kann. Es fällt jedoch auf, dass die Werte währenddessen unter den Wert einer normalen Netzwerkaktivität von 20 PIT-Einträge pro Sekunde fallen.

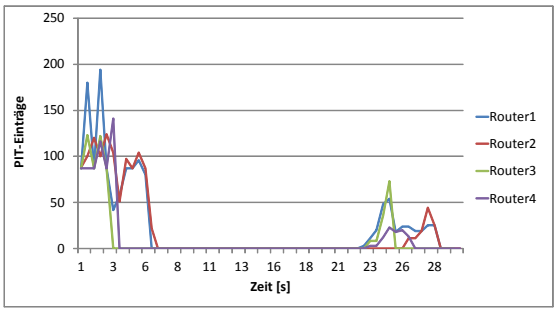

Abbildung 6.46: PIT-Last bei der Interest-Pushback-Methode

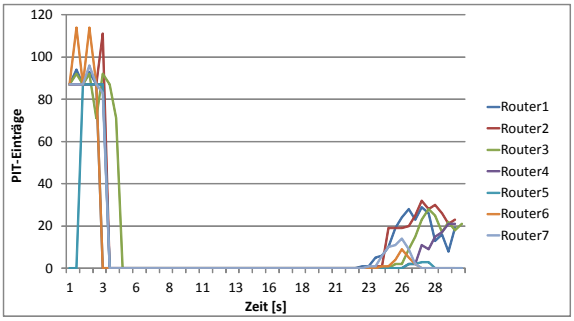

Abbildung 6.47: PIT-Last (Lokal) bei der Interest-Pushback-Methode

Interests-Daten-Verhältnis

In der Abbildung 6.48 wird das Verhältnis von Interests zu Daten pro Sekunde für alle legitimen Consumer dargestellt. Durch eine für legitime Consumer günstige Weiterleitungswahrscheinlichkeit und die nachfolgende Limitierung angreifender Interfaces steigen die Werte bis zum Intervallende auf ca 90%.

Die Abbildung 6.49 zeigt den Verlauf des Verhältnisses in der zweiten Topologie. Die Werte beider Consumer liegen aufgrund des Interface-basierten-Prinzips der Abwehrmethode unter 5%. Bis zum Ende des Intervalls steigen die Werte konstant auf einen Wert bis 30%.

Die Messwerte zeigen, dass das Interests-Daten-Verhältnis der legitimen Consumer aufgrund von topologischen Eigenschaften und der Wirkungsweise der Abwehrmethode zwischen einem Wert von 30%-90% schwanken kann.

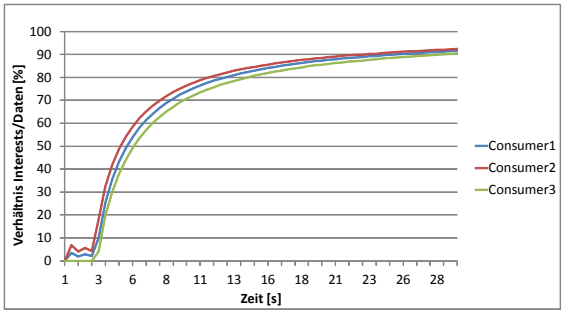

Abbildung 6.48: Interests-Daten-Verhältnis 1 bei der Interest-Pushback-Methode

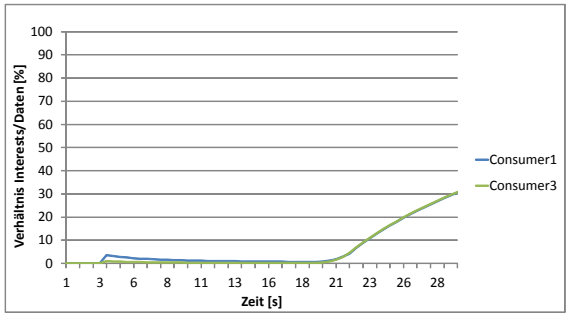

Abbildung 6.49: Interests-Daten-Verhältnis 2 bei der Interest-Pushback-Methode

Content Caching

In der Abbildung 6.50 wird der Verlauf der Datenrate aller legitimer Consumer zur Prüfung des Erhalts der NDN-Eigenschaft Content Caching dargestellt. Nach einer zehnsekündigen Simulation einer normalen Netzwerkaktivität bei rund 148 PIT-Einträgen fallen die Werte zu Beginn des Angriffs auf einen Wert von 0. Anschließend steigt die Datenrate auf rund 160 - 170 und fällt auf den Wert einer normalen Netzwerkaktivität zurück. Somit bleibt mit der Interest-Pushback-Methode durch Abbruch der Datenweiterleitung die Eigenschaft des Content Caching nicht erhalten.

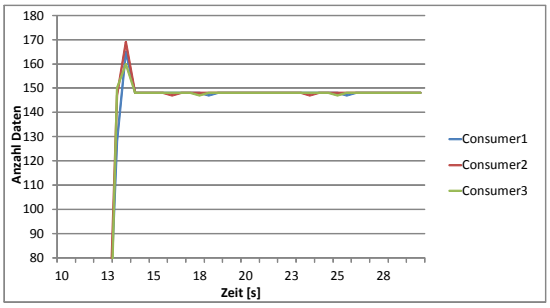

Abbildung 6.50: Content Caching bei der Interest-Pushback-Methode

6.3.10 Interface and Prefix-based Countermeasure

PIT-Last

In der Abbildung 6.51 wird der Verlauf der PIT-Belastung während der Abwehr eines Interest-Flooding-Angriffs mittels der IPC-Methode dargestellt. Durch den Angriff steigen die Werte aller Router – je nach anliegender Angreiferzahl – zunächst stark an, fallen jedoch in der folgenden Sekunde durch die eintretende Abwehr auf einen Normalwert von rund 20 PIT-Einträgen. Da Router4 ausschließlich mit Angreifern verbunden ist, werden alle Präfixe blockiert, wodurch die PIT-Last dieses Routers auf 0 fällt.

Die Abbildung 6.52 zeigt einen ähnlichen Verlauf, wobei erneut die Router auf einen Normalwert sinken. Router5 hingegen blockiert den Präfix des anliegenden Angreifers und fällt auf einen Wert von 0, da er keine weiterleitende Funktion innerhalb des Netzwerkes hat.

Die Messwerte zeigen, dass die Abwehrmethode alle Angreifer und deren Präfixe blockiert, wodurch die PIT-Belastung ohne Kollateralschäden, wie z.B. die Beeinträchtigung legitimer Consumer, gesenkt werden kann.

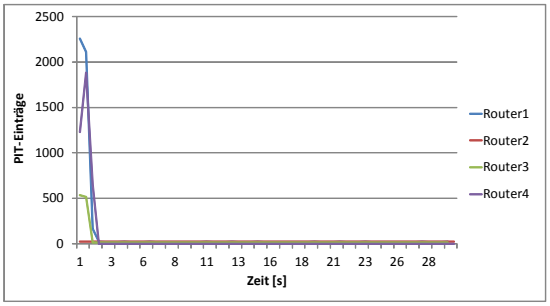

Abbildung 6.51: PIT-Last bei der IPC-Methode

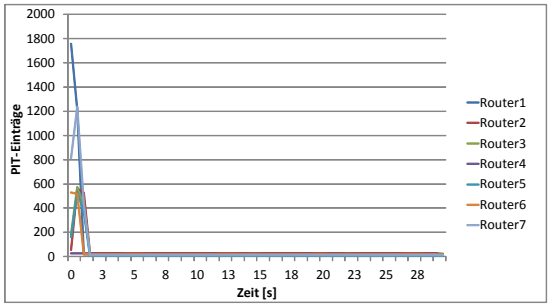

Abbildung 6.52: PIT-Last (Lokal) bei der IPC-Methode

Interests-Daten-Verhältnis

In der Abbildung 6.53 werden die Interests-Daten-Verhältnisse aller legitimen Consumer während der Abwehr gezeigt. Diese steigen innerhalb der ersten Sekunden der Simulation auf einen Wert von über 99% und streben anschließend bis auf 100%.

Die Abbildung 6.54 stellt einen ähnlichen Verlauf dar, wobei die Werte des Verhältnisses trotz topologischen „Flaschenhalses“ gleich hoch bleiben.

Die beiden Abbildungen zeigen deutlich, dass das Interests-Daten-Verhältnis unabhängig von der Topologie während der Abwehr nahezu 100% ist.

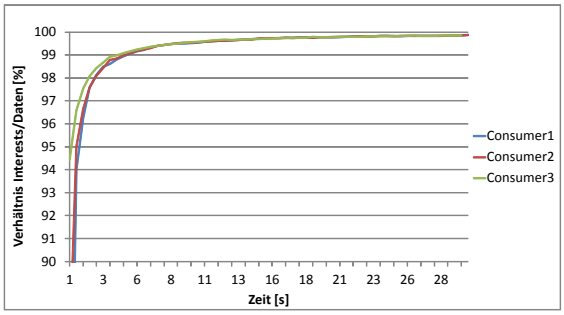

Abbildung 6.53: Interests-Daten-Verhältnis 1 bei der IPC-Methode

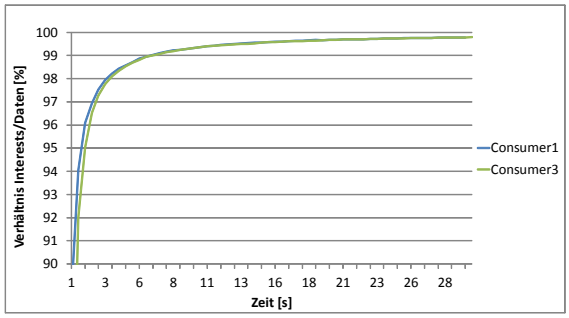

Abbildung 6.54: Interests-Daten-Verhältnis 2 bei der IPC-Methode

Content Caching

In der Abbildung 6.55 wird geprüft, ob die NDN-Eigenschaft des Content Caching während der Abwehr erhalten bleibt. Da die Datenrate trotz einsetzenden Angriffes bzw. Abwehr nahezu konstant bei 148 Daten bleibt, kann davon ausgegangen werden, dass die Eigenschaft erhalten bleibt.

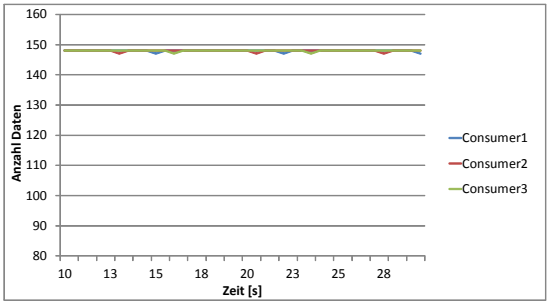

Abbildung 6.55: Content Caching bei der IPC-Methode

6.4 Bewertung

Nachdem die einzelnen Abwehrmethoden innerhalb der Vergleichstopologien getestet wurden, werden nun die erhaltenen Ergebnisse hinsichtlich der Leistungswert in den Tabellen 6.1 und 6.2 dargestellt und anschließend miteinander verglichen.

Methode	**PIT-Last** [mittlere Anzahl der PIT-Einträge]					**Interests-Daten-Verhältnis 1** [%]				**Interests-Daten-Verhältnis 2** [%]		
	R1	R2	R3	R4	**Gesamt**	C1	C2	C3	**Gesamt**	C1	C3	**Gesamt**
IPC	96,1	23,4	32,6	62,8	**53,6**	99	99	99	**99**	99	99	**99**
TDM	151,7	23,4	49,1	100,6	**81,0**	99	99	99	**99**	99	99	**99**
DDoS-Methode	57,4	22,7	15,2	41,0	**34,1**	98	99	96	**98**	97	96	**97**
Poseidon Distributed	36,1	18,4	146,4	28,1	**57,0**	98	98	97	**98**	0	0	**0**
Interest Pushback	23,3	19,9	10,7	11,6	**16,4**	71	73	69	**71**	6	6	**6**
Poseidon Local	36,1	18,4	289,3	170,9	**128,4**	98	98	97	**98**	0	0	**0**
Interest Acceptance	16,6	17,2	15,5	16,0	**16,3**	0	0	10	**3**	0	0	**0**
Interest Traceback	74,4	22,3	24,0	39,0	**39,9**	7	5	10	**7**	3	3	**5**
Token Bucket	126,1	96,3	101,1	112,7	**109,0**	5	3	5	**4**	4	5	**4**
Resource Allocation	1199,5	202,9	1119,5	1200,0	**928,5**	98	98	97	**98**	53	52	**53**

Tabelle 6.1: Zusammenfassung der PIT-Last und der Interests-Daten-Verhältnisse aller Abwehrmethoden

Methode	**Content-Caching** [Anzahl]				**PIT-Last(Lokal)** [Anzahl]	**Einsatzmöglichkeit**
	C1	C2	C3	**Gesamt**	**Gesamt**	
IPC	147,9	147,9	147,9	**147,9**	**40,1**	**überall**
TDM	147,9	147,9	147,9	**147,9**	**59,1**	**überall**
DDoS-Methode	147,9	147,9	147,9	**147,9**	**14,0**	**überall**
Poseidon Distributed	147,9	147,9	147,9	**147,9**	**90,9**	**Netzwerk ohne Flaschenhals (Interface)**
Interest Pushback	130,9	130,6	129,7	**130,4**	**10,6**	**Netzwerk ohne Flaschenhals (Interface)**
Poseidon Local	147,9	147,9	147,9	**147,9**	**171,9**	**Netzwerk ohne Flaschenhals (Interface)**
Interest Acceptance	70,2	32,7	10,4	**37,8**	**15,7**	**Netzwerk ohne Datengüte**
Interest Traceback	3,4	5,7	12,9	**7,4**	**16,0**	**Netzwerk ohne Datengüte**
Token Bucket	9,8	4,7	5,0	**6,5**	**102,3**	**Netzwerk ohne Datengüte**
Resource Allocation	12,7	16,6	18,1	**15,8**	**59,1**	**Netzwerk ohne Datengüte**

Tabelle 6.2: Zusammenfassung für Content Caching, PIT-Last (Lokal) und Einsatzmöglichkeiten

Die Ergebnisse der einzelnen Leistungswerte PIT-Last, Interests-Daten-Verhältnis und Content Caching werden als Einzelwerte der Router bzw. Consumer und als Gesamtwerte in dem jeweiligen Netzwerk angegeben. Dabei wurde die PIT-Last als Durchschnitt der Anzahl der PIT-Einträge, das Interests-Daten-Verhältnis als prozentualer Durchschnitt und das Content Caching als durchschnittliche Anzahl der Daten über die gesamte Simulationszeit für jede Abwehrmethode berechnet.

Im Folgenden werden die einzelnen Abwehrmethoden hinsichtlich ihrer Einsatzmöglichkeiten in Netzwerken bzw. ihrer Wirkungsweise absteigend näher erläutert:

Die Werte der drei Abwehrmethoden DDoS, TDM und IPC zeigen deutlich, dass in allen Fällen durch einen Angriff erzeugte hohe PIT-Einträge auf einen Normalwert gesenkt und die Interest-Daten-Verhältnisse legitimer Consumer gleichzeitig gehalten werden können. Die Eigenschaft des Content Caching bleibt dabei während der Anwendung der Abwehrmetho-

den erhalten. Somit weisen alle Verfahren ein weites Spektrum an Einsatzmöglichkeiten auf. Da es sich doch bei DDoS und TDM um ausschließlich Präfix-basierende Abwehrmethoden handelt, könnten diese bei einem entsprechenden Angriff im Gegensatz zur IPC-Methode schlechter abschneiden.

Die Werte der Poseidon-Distributed-, Interest-Pushback- und Poseidon-Local-Methode zeigen, dass eine Abwehr des Interest-Flooding-Angriffs durch Senkung der künstlich PIT-Einträge stattfindet, wobei die Werte der Poseidon-Local-Methode erhöht sind. Im Gegensatz zu den Präfix-basierenden Methoden, weisen sie jedoch, aufgrund ihrer Pro-Interface-Prüfstelle, deutliche Defizite bei dem Interest-Daten-Verhältnis legitimer Consumer auf, da in Netzwerken mit einem Interface-Flaschenhals (verschiedene Präfixe pro Interface) nicht zwischen Angreifer und Consumer unterschieden werden kann. Die Eigenschaft des Content Caching bleibt in allen drei Fällen erhalten. Somit verfügen die drei Abwehrmethoden aufgrund ihrer Prüfstelle nur über ein eingeschränktes Einsatzspektrum und sind ausschließlich in Netzwerken ohne Interface-Flaschenhals effizient. Die Werte der Abwehrmethoden Interest Acceptance, Interest-Traceback und Token Bucket zeigen, dass ein Angriff durch Senkung der PIT-Einträge abgewehrt werden kann, wobei die Werte der Token-Bucket-Methode erhöht bleiben. Bei dem Interest-Daten-Verhältnis – unabhängig von der verwendeten Topologie – und bei der Eigenschaft des Content Caching fällt jedoch ein starkes Defizit auf. Somit gewährleisten diese Verfahren während der Abwehr keine zufriedenstellende Datenverteilung für die Consumer und somit keine Datengüte, wodurch das Spektrum der Einsatzmöglichkeiten stark begrenzt ist.

Die Werte der Resource-Allocation-Methode zeigen, dass ein Interest-Flooding-Angriff aufgrund des gleichberechtigten Weiterleitungsverhaltens nicht abgewehrt werden kann. Des Weiteren gibt es dadurch Defizite in der Datenverteilung und somit kein befriedigendes Interest-Daten-Verhältnis. Die Eigenschaft des Content Caching bleibt erhalten. Die Einsatzmöglichkeiten dieser Abwehrmethode sind, aufgrund ihres Prinzips sehr stark eingegrenzt, so dass diese ausschließlich von Netzwerken ohne Datengüte und mit ausreichenden Router-Ressourcen genutzt werden kann.

Die vorangegangene Bewertung der Abwehrmethoden lässt folgende Schlussfolgerungen zu:

Die Effizienz einer Abwehrmethode bei einem Interest-Flooding-Angriff steht in enger Verbindung zur verwendeten Prüfstelle. Bei Abwehrmethoden, die auf einer Pro-Router-Prüfstelle basieren, kann, durch die fehlende Lokalisierung des Angriffs, die Datenlieferung an legitime Consumer beeinträchtigt werden. Bei Pro-Interface-basierenden Abwehrmethoden kann – trotz ihrer effektiven PIT-Senkungen – nicht genau bestimmt werden, um welchen Knoten es sich genau handelt, wenn mehrere Knoten hinter einem Interface liegen. Dadurch könnten als Nebeneffekt legitime Consumer limitiert werden.

Da in dieser Arbeit ausschließlich ein Interest-Flooding-Angriff mit gleichbleibendem Präfix betrachtet wird, ist die Effizienz der Pro-Präfix-Methoden bei allen Leistungswerten sehr hoch. Sollte allerdings der betreffende Präfix zeitgleich von legitimen Consumern genutzt werden, so würden diese ebenfalls als Angreifer gesehen.

Dieses und alle zuvor genannten Probleme könnten durch eine Kombination aus einer Pro-Interface- und einer Pro-Präfix-Prüfstelle – wie bei Interface and Prefix-based Countermeasure – behoben werden, da somit ein Angreifer eindeutig identifiziert und Kollateralschäden in Form von Beeinträchtigungen legitimer Consumer verhindert werden könnten.

KAPITEL 7

Zusammenfassung und Aussicht

Named Data Networking ist Gegenstand aktueller Netzwerk-Forschungen und ein möglicher Ansatz für ein Internet mittels Ende-Daten-Konzepts. Durch das Publish-Subscribe-Prinzip bietet es im Gegensatz zum heutigen Internet einen neuen Angriffsvektor auf die Netzwerk-Infrastruktur mittels Interest Flooding Attack. Um einem solchen Angriff effektiv vorzubeugen bzw. dessen Auswirkungen zu minimieren, werden bereits entsprechende Abwehrmethoden entwickelt und getestet.

In der vorliegenden Arbeit wurden sowohl Architektur und Funktionsweise von Named Data Networking als auch das Prinzip der Interest Flooding Attack untersucht. Dazu wurden zunächst in der Literatur beschriebene Abwehrmethoden ausgewählt. Die inhaltliche Beschäftigung mit diesen Anti-Interest-Flooding-Ansätzen machte deutlich, dass deren Angriffserkennung jeweils ausschließlich auf einem Kriterium basiert und somit keine eindeutige Identifizierung des Angreifers ermöglicht wird. Durch die Entwicklung einer Abwehrmethode, die auf zwei Prüfstellen basiert, konnte dies umgesetzt werden. Die Wirksamkeit aller Abwehrmethoden wurde simulativ mit dem ndn-Simulator evaluiert. Hierfür wurde die Implementierung der Abwehrmethoden näher erläutert. Um einen verlässlichen Vergleich der Methoden vornehmen zu können, erfolgte zunächst die Verifikation der Implementierung gegenüber den Originalergebnissen der Verfahren. In einem weiteren Schritt wurden Leistungswerte definiert und beschrieben. Die anschließend ermittelten Messwerte wurden nach Wirkungsweise und Effizienz verglichen und abschließend bewertet.

Zusammenfassend konnte festgestellt werden, dass die Art der verwendeten Prüfstelle (Pro-Router, Pro-Interface, Pro-Präfix) in engem Zusammenhang zur Effizienz einer Abwehrmethode steht. Während Pro-Router-Methoden sich auf die Angriffserkennung beschränken, ermöglichen Pro-Interface- und Pro-Präfix-Methoden zusätzlich eine Angreifer-Rückverfolgung. Eine Kombination der Pro-Interface- und Pro-Präfix-Prüfstellen ermöglicht die erforderliche eindeutige Identifizierung des potenziellen Angreifers.

In dieser Arbeit wurde ausschließlich ein Interest-Flooding-Angriff mit gleichbleibendem Präfix betrachtet. Daher wäre die Wirksamkeit der Abwehrmethoden unter der Bedingung sich ändernder angreifender Präfixe ebenfalls zu untersuchen. Gleichzeitig ist es erforderlich, weitere Angriffserkennungen bzw. Abwehrmöglichkeiten in Betracht zu ziehen. So könnte zum Beispiel folgende Erweiterung der IPC-Methode auf ihre Wirksamkeit bzw. Steigerung der Effizienz hin vertiefend behandelt werden. Die Angriffserkennung ist durch die

Gewichtung der Prüfstellen eher Präfix-basiert. Durch eine zusätzliche Interface-basierende Angriffserkennung könnte eine differenziertere Abwehr ermöglicht werden.

Da es sich bei den bisherigen Untersuchungen um reine Simulationen handelt, wäre ein nächster erforderlicher Schritt eine Implementierung der Abwehrmethoden in einem realen Netzwerk, um deren Wirksamkeit erneut zu testen und die simulierten Messwerte in der Realität zu prüfen, um letztendlich die effektivste Abwehr aus einem weiten Spektrum verschiedenster Methoden auswählen zu können.

Literaturverzeichnis

[1] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, and D. K. Smetters, "Named data networking (ndn) project," October 2010. [Online]. Available: http://www.named-data.net/ndn-proj.pdf

[2] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in Named Data Networking," in *Proc. of IFIP Networking 2013*, May 2013. [Online]. Available: http://networking2013.poly.edu/program-2/

[3] H. Dai, Y. Wang, J. Fan, and B. Liu, "Mitigate ddos attacks in ndn by interest traceback," August 2013. [Online]. Available: http://s-router.cs.tsinghua.edu.cn/~daihuichen/publications/INFOCOM_NOMEM2013_DDoS_Traceback.pdf

[4] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating interest flooding ddos attacks in named data networking," *CoRR*, vol. abs/1303.4823, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1303.html#abs-1303-4823

[5] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, Nov. 1984. [Online]. Available: http://doi.acm.org/10.1145/357401.357402

[6] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *Internet Computing, IEEE*, vol. 7, no. 6, pp. 68–74, 2003.

[7] T.-N. Yoon, "Future internet technical strategies and analysis of best practices," in *Advanced Communication Technology (ICACT), 2013 15th International Conference on*, 2013, pp. 1–1.

[8] *Named Data Networking*, Nov. 2013. [Online]. Available: http://named-data.net/

[9] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[10] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, April 2004.

[11] N. Fotiou, G. Marias, and G. Polyzos, "Publish–subscribe internetworking security aspects," in *Trustworthy Internet*, L. Salgarelli, G. Bianchi, and N. Blefari-Melazzi, Eds. Springer Milan, 2011, pp. 3–15. [Online]. Available: http://dx.doi.org/10.1007/978-88-470-1818-1_1

[12] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, "Backscatter from the data plane — threats to stability and security in information-centric networking," *CoRR*, vol. abs/1205.4778, 2012.

[13] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, "Backscatter from the data plane–threats to stability and security in information-centric network infrastructure," *Computer Networks*, vol. 57, no. 16, pp. 3192–3206, 2013.

[14] S. Choi, K. Kim, S. Kim, and B. hee Roh, "Threat of dos by interest flooding attack in content-centric networking," *The International Conference on Information Networking 2013 (ICOIN)*, vol. 0, pp. 315–319, 2013.

[15] K. Wang, J. Chen, H. Zhou, Y. Qin, and H. Zhang, "Modeling denial-of-service against pending interest table in named data networking," *International Journal of Communication Systems*, pp. n/a–n/a, 2013. [Online]. Available: http://dx.doi.org/10.1002/dac.2618

[16] E. U. L. Z. P. Gasti, G. Tsudik, "Dos and ddos in named-data networking," in *The 22nd International Conference on Computer Communications and Networks*, 2013.

[17] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN, Technical Report NDN-0005, October 2012. [Online]. Available: http://named-data.net/techreports.html

[18] *Information-Centric Networking Research Group*, Nov. 2013. [Online]. Available: http://irtf.org/icnrg

[19] D. Trossen, M. Sarela, and K. Sollins, "Arguments for an Information-Centric Internetworking Architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 2, pp. 26–33, 2010.

[20] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking." *IEEE Communications Magazine*, 2012.

[21] S. Arianfar, P. Nikander, and J. Ott, "On Content-Centric Router Design and Implications," in *Proc. of ReARCH workshop*. New York, NY, USA: ACM, 2010.

[22] D. Goergen, T. Cholez, J. François, and T. Engel, "Security monitoring for content-centric networking." in *DPM/SETOP*, ser. Lecture Notes in Computer

Science, R. D. Pietro, J. Herranz, E. Damiani, and R. State, Eds., vol. 7731. Springer, 2012, pp. 274–286. [Online]. Available: http://dblp.uni-trier.de/db/conf/esorics/lncs7731.html#GoergenCFE12

[23] K. Wang, H. Zhou, H. Luo, J. Guan, Y. Qin, and H. Zhang, "Detecting and mitigating interest flooding attacks in content-centric network," *Security and Communication Networks*, pp. n/a–n/a, 2013. [Online]. Available: http://dx.doi.org/10.1002/sec.770