TCP-Verkehrsklassifizierung mit Markov-Modellen

Hui Dai, Gerhard Münz, Lothar Braun, Georg Carle
Lehrstuhl für Netzarchitekturen und Netzdienste
Institut für Informatik
Technische Universität München
Email: dai@in.tum.de, {muenz|braun|carle}@net.in.tum.de

Zusammenfassung—**Die Klassifizierung von Verkehrsströmen anhand von Portnummern ermöglicht heutzutage keine zuverlässigen Aussagen bezüglich der ursächlichen Anwendungen. Auch die Verwendung von anwendungsspezifischen Signaturen stößt an ihre Grenzen, sobald der Verkehr verschlüsselt ist. Ein aktuelles Forschungsgebiet beschäftigt sich daher mit Verfahren, die es ermöglichen, Verkehrsströme mit Hilfe von anderen Flow- und Paketeigenschaften einer Anwendung zuordnen zu können.**

In diesem Kontext stellen wir ein Klassifizierungsverfahren vor, das die Abfolge von gesetzten TCP-Flags und Paketgrößen innerhalb einer TCP-Verbindung mit Hilfe von Markov-Modellen modelliert. Stehen Markov-Modelle für verschiedene Anwendungen zur Verfügung, kann eine neue TCP-Verbindung durch Bestimmung der maximalen A-posteriori-Wahrscheinlichkeiten einer dieser Anwendungen zugeordnet werden. Ein Vergleich des Verfahrens mit dem viel zitierten Ansatz von Bernaille [1] zeigt, dass sich mit unserem recht einfachen Ansatz ähnlich gute und bei geeigneter Parametrisierung sogar bessere Klassifizierungsergebnisse erzielen lassen.

I. Einleitung

Eine möglichst genaue Klassifizierung des Internet-Verkehrs nach Anwendungen bzw. Anwendungsklassen ist für Netzbetreiber aus mehrfacher Hinsicht von Interesse. Sie ermöglicht beispielsweise, den Verkehr in verschiedene Dienstgüteklassen einzuteilen, anomale oder bösartige Verkehrsströme zu erkennen oder Informationen über die Nutzung des Netzes zu erheben, woraus sich Schlüsse für die zukünftige Netzplanung ziehen lassen. Früher war die Klassifizierung des Internet-Verkehrs relativ einfach, da sich der Verkehr im Wesentlichen aus Anwendungen wie Web, FTP, SMTP und Telnet zusammensetzte, deren Protokolle bei der Internet Assigned Numbers Authority (IANA) mit unterschiedlichen, so genannten „well-known“ oder „registered“ TCP- und UDP-Portnummern registriert sind. Von der Verwendung dieser vorgegebenen Portnummern wurde im Allgemeinen nicht abgewichen, sodass die Anwendung anhand des entsprechenden Paket-Header-Feldes identifiziert werden konnte.

Heutzutage nimmt die Klassifizierungsgenauigkeit, die mit diesem Verfahren erreicht werden kann, immer weiter ab, weil viele neue Protokolle keinen bei der IANA registrierten Port verwenden. Außerdem gibt es neue Anwendungen, die die Portnummern dynamisch festlegen, wie z. B. Skype. Viele Peer-to-Peer-Anwendungen (P2P) verwenden die Well-Known-Ports anderer Anwendungen, um Firewall-Regeln zu umgehen. In einer Arbeit von 2005 konnte anhand der Portnummer ein Anteil von 69% bzw. 71% des Verkehrsvolumens (gemessen in Bytes bzw. Paketen) in einem Campus-Netz in verschiedene Anwendungsklassen korrekt eingeordnet werden [2]. In einer Studie von 2008 konnten auf diese Weise je nach Netzwerk und Verkehrszusammensetzung 71% und 95% der Flows einer Anwendung korrekt zugeordnet werden [3]. Ein Flow ist hierbei wie üblich durch Transportprotokoll, Quell- und Zieladresse sowie Quell- und Zielport definiert. Für klassische Anwendungen wie Web, Mail, DNS, Chat und SSH liegt die Genauigkeit deutlich über 90%, für FTP, P2P, Streaming-Anwendungen und Online-Spiele teilweise weit unter diesem Wert.

Ein anderes klassisches Verfahren untersucht die Paketinhalte des Internet-Verkehrs. Eine große technische Herausforderung ist hierbei, mit den immer höheren Paketraten im Internet umgehen zu können. Daneben nimmt mit der Zahl der Anwendungen auch die Zahl der Signaturen stetig zu, wodurch die Klassifizierung mehr Speicherplatz und einen höheren Rechenaufwand je Paket benötigt. Die Notwendigkeit zusätzlicher Signaturen kann sich auch schon bei einer neuen Version einer Anwendung ergeben. Ein hoher Anpassungsbedarf ergibt sich dadurch, dass die verwendeten Signaturen laufend auf dem neusten Stand gehalten werden müssen. Bei verschlüsseltem Internet-Verkehr ist zudem eine inhaltsbasierte Klassifizierung gar nicht möglich.

Neben den technischen Problemen müssen bei der Untersuchung von Paketinhalten auch rechtliche Aspekte berücksichtigt werden. Aus datenschutzrechtlichen Gründen ist es häufig nicht zulässig, den Inhalt des Verkehrs zu analysieren und zu kontrollieren.

Angesichts der Einschränkungen der port- und inhaltsbasierten Klassifizierung versuchen seit einigen Jahren verschiedene Wissenschaftler, neue Methoden zu finden, die auf statistischen Verkehrseigenschaften beruhen. Zum großen Teil werden Methoden des maschinellen Lernens (ML) angewandt, wie z. B. Clustering-Algorithmen oder naive Bayes-Klassifikatoren [4]. Einige wenige Arbeiten verwendeten auch Hidden-Markov-Modelle. In Abschnitt II werden wir die wichtigsten Ansätze kurz vorstellen.

Im Anschluss stellen wir in Abschnitt III ein neues Verfahren vor, das Markov-Modelle zur Klassifizierung von TCP-Verbindungen verwendet. Die betrachteten Eigenschaften sind die Längen der Nutzdaten und die TCP-Flags der ersten Pakete einer TCP-Verbindung. Ein Vergleich mit dem viel zitierten Ansatz von Bernaille [1], [5] zeigt, dass unser Verfahren ähnlich gute und teilweise sogar bessere Ergebnisse erzielt. Dabei ist unser Verfahren einfacher zu implementieren, weni-

ger rechenaufwendig und leichter zu parametrisieren. In Abschnitt IV präsentieren wir diese Untersuchungsergebnisse, die auf Verkehrsdaten basieren, die in echten Netzen gesammelt wurden.

Abschnitt V schließt unsere Ausführungen mit einer Diskussion der Ergebnisse und einem Ausblick auf mögliche Verbesserungen des Verfahrens ab.

II. Stand der Forschung

Für die Verkehrsklassifizierung wurden mehrfach ML-Techniken vorgeschlagen, die in Trainingsdaten anwendungsspezifische Muster und Unterscheidungsmerkmale finden sollen. Die betrachteten Eigenschaften beziehen sich dabei entweder auf den gesamten Flow, wie z.B die mittlere Paketlänge, oder auf einzelne Pakete innerhalb eines Flows oder einer Verbindung, wie z.B. die Paketgröße oder der zeitliche Abstand zum vorangegangenen Paket (Zwischenankunftszeit). Nguyen und Armitage liefern einen ausführlichen Vergleich dieser Verfahren [4].

Im Folgenden gehen wir auf Arbeiten ein, die ähnlich wie unser Ansatz Markov-Modelle verwenden oder sich auf die Untersuchung der ersten Pakete einer Verbindung bzw. eines Flows beschränken, ohne dabei die übertragenen Nutzdaten zu betrachten.

Bernaille et al. klassifizieren TCP-Verkehr anhand der Längen der ersten vier bis fünf Datenpakete einer Verbindung [1], [5]. Datenpakete sind dabei diejenigen Pakete, die Nutzdaten enthalten. Zur Klassifizierung werden die unüberwachten (engl. „unsupervised“) Cluster-Algorithmen k-Means und GMM (Gaussian Mixture Models) verwendet, obgleich die Trainingsdaten nach Anwendungen markiert sind (engl. „labeled“) und von daher auch überwachte (engl. „supervised“) ML-Methoden verwendet werden könnten. Die gefundenen Cluster werden anschließend derjenigen Anwendung zugeteilt, aus der die Mehrzahl der zugeordneten Verbindungen stammt. Die Autoren sprechen in diesem Zusammenhang von halbüberwachtem (engl. „semi-supervised“) Lernen. Bei der Klassifizierung einer neuen Verbindung wird deren Cluster-Zugehörigkeit bestimmt und die entsprechende Anwendung angenommen. Um das Ergebnis zu verbessern, präsentieren Bernaille et al. auch eine Variante ihres Verfahrens, die neben der Cluster-Zugehörigkeit die Portnummern mitberücksichtigt [1].

Abgesehen von k-Means und GMM untersuchen Bernaille et al. auch den Einsatz von *Hidden-Markov-Modellen* (HMM) und so genannten spektralen Clustering-Methoden [1], [6]. Jede Verbindung wird dabei in ein Links-Rechts-HMM überführt, zwischen denen die *Log-Likelihood* als Ähnlichkeitsmaß berechnet werden kann. Um ein gutes HMM aus einer einzigen Sequenz zu bestimmen, muss die Sequenz sehr viel länger als die Zustandsanzahl sein [7], [8]. Sofern wir Bernaille et al. nicht missverstehen [1], [6], ordnen sie aber jeden Zustand genau einem Paket zu, was bedeutet, dass die Anzahl der Pakete einer Verbindung gleich der Anzahl der Zustände des zugehörigen HMM wäre. Die Folge wären deterministische Modelle, für die die Berechnung der Log-Likelihood keinen Sinn macht. Es bleibt also unklar, inwieweit die Autoren HMMs sinnvoll eingesetzt haben.

Wright et al. [9] sowie Dainotti et al. [10] bestimmen mit Hilfe von Trainingsdaten je ein HMM für jede Anwendung. Der Klassifizierer ordnet dann eine gegebene Paketfolge derjenigen Anwendung zu, für die das HMM die größte Log-Likelihood ergibt. Beide Forschergruppen modellieren dabei nicht nur die Paketlängen sondern auch die Zwischenankunftszeiten. Wright et al. betrachten beide Richtungen von TCP-Verbindungen und verwenden Links-Rechts-HMMs mit einer großen Zustandsanzahl und einer diskreten Emissionswahrscheinlichkeitsverteilung. Dainotti et al. verwenden dagegen ergodische HMMs mit vier bis sieben Zuständen und gammaverteilten Emissionswahrscheinlichkeiten. Betrachtet werden nur Datenpakete (d.h. Pakete mit Nutzdaten), die vom Client zum Server fließen.

Im Gegensatz zu den genannten Arbeiten verwenden wir keine Markov-Modelle mit versteckten Zuständen sondern solche, bei denen die Zustände mit den Beobachtungen übereinstimmen. Motiviert wurden wir dabei von Estevez-Tapiador et al., die Markov-Modelle zur Erkennung von Anomalien in TCP-Verbindungen verwendet haben [11]. Die Zustände leiten sich hier aus den TCP-Flags der Pakete eines Flows ab, ein Zustandsübergang entspricht einer neuen Paketankunft. In der Trainingsphase wird aus Verkehrsdaten für verschiedene Anwendungen ein Modell für normalen Verkehr ohne Angriffe gebildet. In der Erkennungsphase wird für jede beobachtete TCP-Verbindung die Auftrittswahrscheinlichkeit (engl. „maximum a-posteriori probability“) unter der Annahme berechnet, dass das entsprechende Markov-Modell gültig ist. Für anomalen Verkehr fällt diese Wahrscheinlichkeit deutlich kleiner aus als für normalen Verkehr, was sich zur Erkennung von Anomalien und Angriffen ausnutzen lässt.

Wie im folgenden Abschnitt beschrieben, verwenden wir einen ähnlichen Ansatz zur Verkehrsklassifizierung. Dabei betrachten wir nicht nur TCP-Flags sondern auch Paketgrößen.

III. TCP-Verbindungsklassifizierung mit Markov-Modellen

Wie alle Arbeiten, die Verkehrsklassifizierung anhand statistischer Eigenschaften des Netzwerkverkehrs betreiben, gehen auch wir davon aus, dass bestimmte Eigenschaften des Netzverkehrs vom Verhalten der Anwendungen bestimmt werden, die den Verkehr erzeugen. Aus der Beobachtung von Verkehrsströmen können somit Rückschlüsse auf die Anwendungen gezogen werden. Wir modellieren die Eigenschaften von TCP-Verbindungen mit Hilfe von Markov-Modellen, deren Zustände direkt aus den beobachteten Paketen abgeleitet werden. Die betrachteten Paketeigenschaften sind TCP-Flags und Paketlängen.

Das Markov-Modell besteht aus einer Menge von n möglichen Zuständen $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$, einem Vektor für die Anfangswahrscheinlichkeiten $\Pi = (\pi_1, \ldots, \pi_n)$ und einer $n \times n$-Matrix der Übergangswahrscheinlichkeiten $A = \{a_{s_i,s_j}\}$. Übergänge zwischen den verschiedenen Zuständen werden

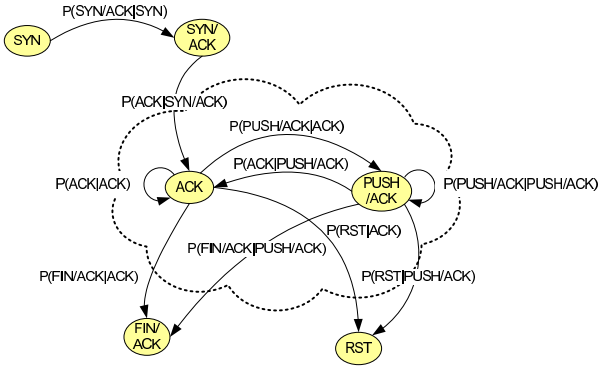
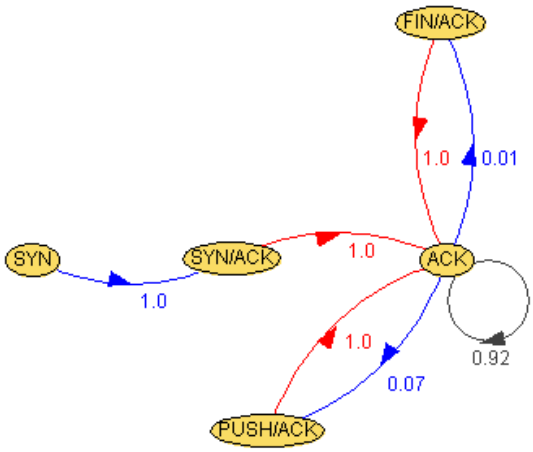
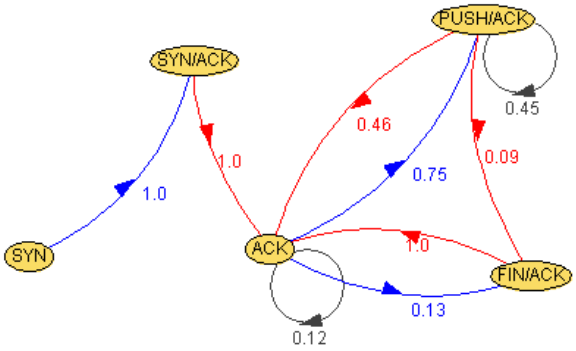


Abb. 1: Markov-Modell mit TCP-Flags als Zustände



(a) HTTP



(b) Gnutella

Abb. 2: Markov-Modelle für TCP-Flags

durch Paketankünfte ausgelöst. Der neue Zustand kann direkt aus den gesetzten TCP-Flags oder der Paketgröße des neu eingetroffenen Paketes abgeleitet werden.

In einem ersten Ansatz betrachten wir nur die TCP-Flags, die in einem Paket gesetzt sind. Je nach dem, welche der Flags URG, ACK, PSH, RST, SYN und FIN gesetzt sind, wird das Paket einem Zustand zugeordnet. Die 64 möglichen TCP-Flag-Kombinationen werden also auf 64 verschiedene Zustände abgebildet, wovon in einer typischen TCP-Verbindung aber nur ein kleiner Teil auftritt. Der übliche Ablauf einer TCP-Verbindung folgt dem Markov-Modell, das in Abbildung 1 dargestellt ist. Der Aufbau einer TCP-Verbindung durch den sogenannten *Three-Way-Handshake* ist für alle Anwendungen gleich. Gleiches gilt im Allgemeinen auch für den Verbindungsabbau, wobei hier gewisse Formen des Verbindungsabbruchs charakteristisch für eine Anwendung sein können. Interessant für die Unterscheidung verschiedener Anwendungen sind vor allem die TCP-Flags in den Paketen, die zwischen Verbindungsaufbau und -abbau ausgetauscht werden, also das Auftreten von ACK- und ACK/PUSH-Paketen.

In einem zweiten Ansatz betrachten wir nur die Paketlängen innerhalb einer TCP-Verbindung. Die Paketlänge ist nach unten durch den TCP/IP-Header und nach oben durch die MTU (*maximum transmission unit*) begrenzt. Es macht nun keinen Sinn, jede mögliche Paketlänge auf einen diskreten Zustand abzubilden, weil die Auftrittswahrscheinlichkeiten sehr gering wären und der Zustandsraum sehr groß. Deshalb unterteilen wir die Menge der möglichen Paketlängen in Intervalle, die jeweils einen Zustand bilden.

Weiterhin ist es möglich, Zustände zu erzeugen, die sich aus einer Kombination aus TCP-Flags und Paketlänge ergeben. Eine solche Kombination könnte die Genauigkeit der Klassifikation gegenüber der Verwendung einer einzelnen Paketeigenschaft verbessern. Zusätzlich kann auch die Flussrichtung berücksichtigt werden, d.h. Pakete, die vom Client zum Server fließen, werden auf andere Zustände abgebildet als Pakete in entgegengesetzter Richtung. Als Client verstehen wir hierbei stets den Kommunikationspartner, der die TCP-Verbindung initiiert hat. In Abschnitt IV untersuchen wir die verschiedenen Möglichkeiten, die Paketeigenschaften auf die Zustände des Markov-Modells abzubilden.

Nachdem die Zustände definiert sind, können die Anfangswahrscheinlichkeiten π_i und die Übergangswahrscheinlichkeiten a_{s_i,s_j} direkt aus Trainingsdaten geschätzt werden. Die Anfangswahrscheinlichkeiten ergeben sich aus den relativen Häufigkeiten der modellierten Eigenschaft in den ersten Paketen. Für die Übergangswahrscheinlichkeiten ermittelt man die Häufigkeiten der verschiedenen Zustandsübergänge $F(s_i, s_j)$ mit $s_i, s_j \in \Sigma$ in den TCP-Verbindungen einer Anwendung. Die Übergangswahrscheinlichkeit für den Zustandswechsel von s_i nach s_j ergibt sich dann wie folgt:

$$a_{s_i,s_j} = \frac{F(s_i, s_j)}{\sum_{k=1}^{n} F(s_i, s_k)} \quad (1)$$

Somit erhält man für jede Anwendung ein eigenes Markov-Modell. Die Abbildungen 2a und 2b zeigen beispielhaft die unterschiedlichen Markov-Modelle für die TCP-Flags, die sich aus den Trainingsdaten für Webverkehr (HTTP) und Gnutella ergeben. Der SYN-Zustand ist hier der einzig mögliche Anfangszustand, da alle Verbindungen mit einem SYN-Paket beginnen.

Wie in Abbildung 3 dargestellt werden die aus Trainingsdaten erzeugten Modelle in der Klassifizierungsphase dazu ein-

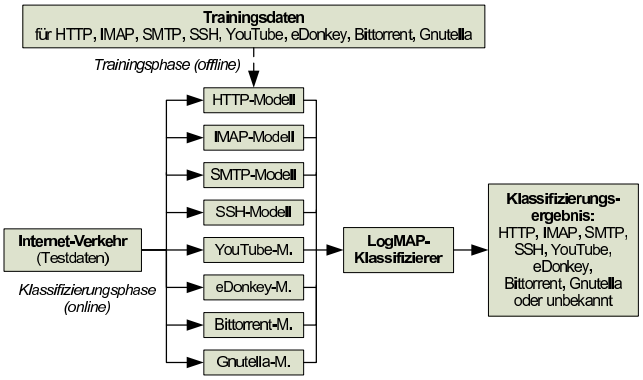


Abb. 3: Schematische Darstellung der Klassifizierung

gesetzt, neue TCP-Verbindungen einer Anwendung zuzuordnen. Dazu werden für die Paketfolge einer Verbindung die maximale A-posteriori-Wahrscheinlichkeiten (engl. „maximum a-posteriori probabilities“) aller Markov-Modelle berechnet. Die maximale A-posteriori-Wahrscheinlichkeit für ein Modell M ist die Wahrscheinlichkeit für das Auftreten der Paketfolge $O = \{o_1, o_2, \ldots, o_N\}$ unter der Annahme, dass das Modell gilt:

$$MAP(O, M) = \Pr(O|M) = \pi_{o_1} \prod_{i=1}^{N-1} a_{o_i, o_{i+1}} \qquad (2)$$

Da die Wahrscheinlichkeiten sehr klein werden können, ist es nahe liegend zu logarithmieren:

$$LogMAP(O, M) = \log \pi_{o_1} + \sum_{i=1}^{N-1} \log a_{o_i, o_{i+1}} \qquad (3)$$

Da alle TCP-Verbindungen mit einem SYN-Paket beginnen, sind sowohl das Start-Flag aus auch die erste Paketlänge bekannt und für jede Verbindung identisch. Daher gilt $\pi_{o_1} = 1$ für die Anfangswahrscheinlichkeit. Die Berechnung vereinfacht sich somit zu

$$LogMAP\,(O, M) = \sum_{i=1}^{N-1} \log a_{o_i, o_{i+1}} \qquad (4)$$

Der Klassifizierer wählt nun dasjenige Modell aus, für das die logarithmierte A-posteriori-Wahrscheinlichkeit maximal ist.

Es kann der Fall auftreten, dass ein Übergang o_i, o_{i+1} in der zu klassifizierenden TCP-Verbindung vorkommt, der in den Trainingsdaten einer Anwendung nicht auftrat. In diesem Fall ist die modellierte Übergangswahrscheinlichkeit $a_{o_i, o_{i+1}} = 0$, d.h. die A-posteriori-Wahrscheinlichkeit wird ebenfalls Null und das Markov-Modell ist nicht passend. Falls der Übergang in den Markov-Modellen sämtlicher Anwendungen nicht vorkommt, kann die Verbindung keiner Anwendung zugeordnet werden („Unknown“ in Abbildung 3).

In Abbildung 4 sind Beispiele für $LogMAP$-Diagramme zu sehen, die den Verlauf von $LogMAP$ in Abhängigkeit von der Anzahl der Pakete N aufzeigen (vgl. Gleichung 4). Jeder Zustand in den Markov-Modellen steht hierbei für eine bestimmte TCP-Flag-Kombination und eine bestimmte Flussrichtung. In Abbildung 4a ist für eine HTTP-Verbindung und eine SSH-Verbindung der Verlauf von $LogMAP$ aufgetragen, der sich für ein Markov-Modell ergibt, das aus HTTP-Trainingsdaten erzeugt wurde. Wie man sieht sinkt $LogMAP$ im Fall von SSH mit zunehmender Paketanzahl sehr viel schneller, was zeigt, dass das HTTP-Modell für SSH-Verbindungen schlechter passt als für HTTP-Verbindungen. Der Netzverkehr beider Anwendungen unterscheidet sich offentsichlich bezüglich des Auftretens der TCP-Flags.

Abbildung 4b verdeutlicht das Vorgehen bei der Klassifizierung anhand einer HTTP-Verbindung. Hier ist der Verlauf von $LogMAP$ für die Markov-Modelle verschiedener Anwendungen aufgetragen. Dabei zeigt sich, dass das HTTP-Modell tatsächlich am besten zu der Verbindung passt. Diese Feststellung kann bereits nach wenigen Paketen getroffen werden. Für die Klassifikation muss daher nicht bis zum Ende der Verbindung gewartet werden, was eine wichtige Voraussetzung für die Online-Verkehrsklassifizierung darstellt. Interessant ist, dass das YouTube-Modell am zweitbesten abschneidet. Die Ähnlichkeit zwischen den Markov-Modellen der beiden Anwendungen lässt sich dadurch erklären, dass bei YouTube Videos über HTTP übertragen werden.

Im folgenden Abschnitt evaluieren wir das vorgestellte Verfahren und untersuchen dabei verschiedene Möglichkeiten, TCP-Flags und Paketgrößen auf die Zustände der Markov-Modelle abzubilden. Zudem vergleichen wir das Verfahren mit dem viel zitierten Verfahren von Bernaille [1].

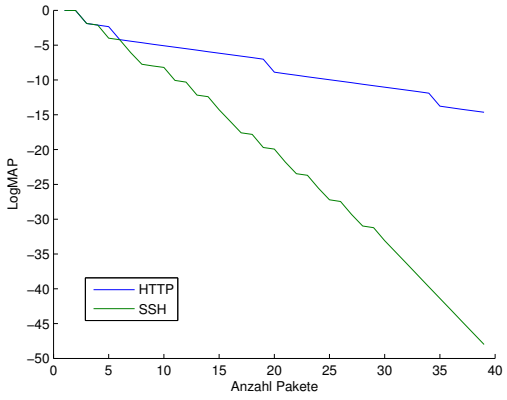
IV. Evaluierung und Vergleich

Die Wirksamkeit des vorgestellten Klassifizierungsansatzes haben wir anhand von Verkehrsdaten evaluiert, die in echten Netzen aufgezeichnet wurden. Wo dies geschah und wie die Daten vorverarbeitet wurden, wird im folgenden Unterabschnitt beschrieben. Danach folgen in den Unterabschnitten IV-C, IV-D und IV-E Untersuchungen, bei denen in den Zuständen der Markov-Modelle TCP-Flags, Paketgrößen oder beide Paketeigenschaften berücksichtigt werden. In Unterabschnitt IV-F vergleichen wir unseren Ansatz mit dem von Bernaille [1]. Unterabschnitt IV-G liefert schließlich noch Angaben zum Klassifizierungsaufwand.

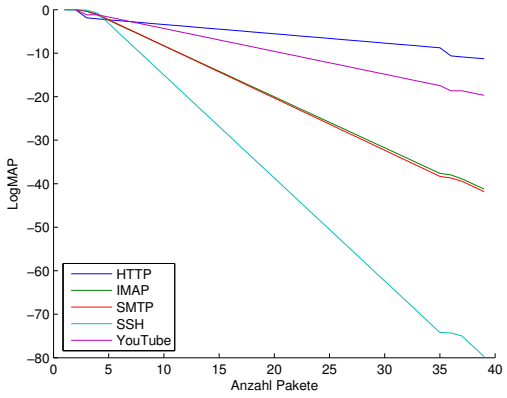
A. Trainings- und Testdaten

Die für die Evaluierung verwendeten Verkehrsaufnahmen wurden in den Netzwerken unserer Arbeitsgruppe an der Universität Tübingen und der Technischen Universität München gesammelt. Sie umfassen zum einen Daten klassischer Client-Server-Protokolle wie HTTP, IMAP, SMTP und SSH. Dieser Verkehr wurde an unseren eigenen Servern aufgenommen, sodass sichergestellt ist, dass die Messdaten der richtigen Anwendung zugeordnet wurden. Weitere Aufnahmen umfassen Verkehr zum Video-Portal YouTube[1], der von uns durch Aufrufen von YouTube-Seiten erzeugt wurde. Da die Verkehrsklassifizierung im Speziellen für Dienste notwendig ist, die

[1] http://www.youtube.com



(a) $LogMAP$ einer SSH-Verbindung und einer HTTP-Verbindung bzgl. eines HTTP-Modells



(b) $LogMAP$ einer HTTP-Verbindung bzgl. verschiedener Modelle

Abb. 4: $LogMAP$-Diagramme

Tabelle I: Zusammensetzung der Trainings- und Testdaten

	Training		Test	
	Verbindungen	Volumen	Verbindungen	Volumen
eDonkey	300	95,4MB	526	181MB
BitTorrent	300	45,5MB	462	42,3MB
Gnutella	300	62,6MB	312	81,4MB
HTTP	300	15,3MB	761	12MB
IMAP	300	5,44MB	595	8,91MB
SMTP	300	3,23MB	599	6,52MB
SSH	300	8,46MB	377	4,36MB
YouTube	300	21,6MB	311	28,1MB

unbekannte bzw. dynamisch ausgehandelte Ports verwenden, untersuchen wir auch Verkehr aus Peer-to-Peer-Netzwerken. Durch eine Installation der entsprechenden Anwendungen auf unseren Rechnern, waren wir in der Lage Verkehr aus den Peer-to-Peer-Netzen eDonkey, BitTorrent und Gnutella zu generieren und aufzuzeichnen.

Tabelle I fasst die Zusammensetzung der Trainings- und Testdaten zusammen. In der Trainingsphase werden von jeder Anwendung 300 Verbindungen verwendet, um jeweils ein Markov-Modell zu erzeugen. Sowohl die Test- als auch die Trainingsdaten enthalten nur Verbindungen mit mindestens

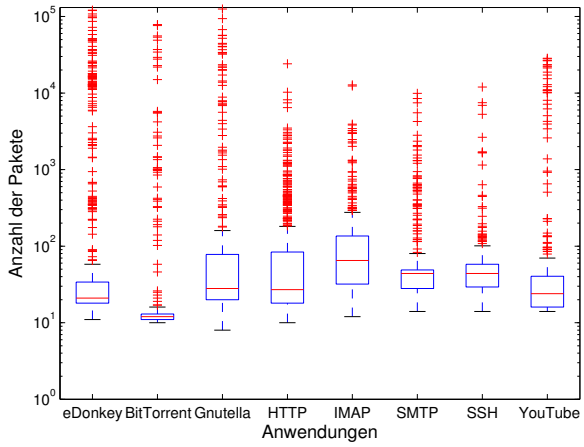


Abb. 5: Paketanzahl pro Verbindung nach Anwendungen

acht Paketen. Durch das Herausfiltern kürzerer Verbindungen wird sichergestellt, dass nur vollständig aufgebaute Verbindungen enthalten sind, die nicht direkt nach Verbindungsaufbau abgebrochen oder beendet wurden. Zudem wurden Verbindungen, bei denen weniger als vier Pakete Nutzdaten enthielten, verworfen, um einen fairen Vergleich mit dem Ansatz von Bernaille zu ermöglichen [1]. Wie in Abschnitt II erwähnt, benötigt Bernaille eine solche Mindestanzahl an Datenpaketen für die Verkehrsklassifzierung. Es sei an dieser Stelle hervorgehoben, dass unser Ansatz auch mit weniger Datenpaketen arbeiten kann.

Abbildung 5 veranschaulicht mit Hilfe von Box-Plots die Verteilung der Paketanzahl je Verbindung für die verschiedenen Anwendungen. Grundlage hierfür sind die in den Trainingsdaten vorhandenen Verbindungen. Man erkennt, dass die meisten Verbindungen zwischen zehn und 100 Pakete pro Verbindung enthalten. Bei der Bildung von Markov-Modellen und der anschließenden Klassifizierung beschränken wir uns auf eine vorgegebene Maximalanzahl von Paketen, die pro Verbindung betrachtet werden. Wie wir später zeigen werden, hat diese Beschränkung Einfluss auf die Klassifikationsgenauigkeit. Andererseits ermöglicht sie uns bei langen Verbindungen ein Klassifikationsergebnis zu erhalten, bevor das Ende einer Verbindung erreicht ist, was eine wichtige Voraussetzung für die Online-Klassifizierung darstellt.

Da die ersten drei Pakete jeder Verbindung zum TCP-Handshake gehören, ist eine Differenzierung verschiedener Anwendungen erst mit Hilfe der darauf folgenden Pakete möglich. In unserer Auswertung verwenden wir deshalb 10 als kleinste Maximalanzahl an berücksichtigten Paketen je Verbindung. Eine Maximalanzahl von 10 bedeutet, dass bei Verbindungen mit zehn Paketen oder weniger alle Pakete bei der Modellbildung und der Klassifizierung berücksichtigt werden, bei Verbindungen mit mehr als zehn Paketen nur die ersten zehn. Bei einer höheren Maximalanzahl werden entsprechend mehr Pakete am Anfang einer jeden Verbindung betrachtet, wobei sich aber auch der Anteil der Verbindungen,

die bis zum Verbindungsende untersucht werden, erhöht. Beispielsweise sieht man an den Boxplots in Abbildung 5, dass bei einer Maximalanzahl von 20 Paketen bereits ein Großteil der BitTorrent-Verbindung vollständig untersucht wird.

B. Bewertungskriterien

Unsere Klassifikationsergebnisse bewerten wir anhand von Trefferquote und Genauigkeit und verwenden hierfür die üblichen englischen Begriffe *Recall* und *Precision*. Recall und Precision sind wie folgt definiert:

- *Recall:* Anteil aller Verbindungen der Anwendung A, die korrekt als Anwendung A klassifiziert werden:

$$\text{Recall} = \frac{\#(\text{korrekt als A klassifizierte Verbindungen})}{\#(\text{alle Verbindungen von A})}$$

- *Precision:* Anteil der korrekt als Anwendung A klassifizierten Verbindungen bezüglich aller als A klassifizierten Verbindungen:

$$\text{Precision} = \frac{\#(\text{korrekt als A klassifizierte Verbindungen})}{\#(\text{alle als A klassifizierte Verbindungen})}$$

Ziel der Klassifizierung ist es, sowohl für Recall als auch für Precision einen Wert nahe 100 Prozent zu erreichen.

Für die Berechnung dieser beiden Gütemaße wird zwischen korrekt und falsch klassifizierten Verbindungen unterschieden. Unter die falsch klassifizierten Verbindungen fallen solche, die einer falschen Anwendung zugeordnet werden, und solche, die keiner Anwendung zugeordnet werden können (vgl. Abschnitt III). Da die Testdaten keine unbekannten Anwendungen enthalten, gelten in unserem Fall sämtliche Verbindungen, die keiner Anwendung zugeordnet werden können, als falsch klassifiziert.

Um die Qualität eines Verkehrsklassifizierungsverfahrens an einem einzigen Wert festzumachen, wird in den verwandten Arbeiten häufig die Klassifizierungsgenauigkeit für sämtliche Verbindungen der Testdaten bestimmt (engl. *overall accuracy*), also die relative Häufigkeit, mit der die Verbindungen korrekt klassifiziert wurden. Diese Genauigkeit hängt maßgeblich von der Verkehrszusammensetzung ab. Enthalten die Testdaten viele Verbindungen von Anwendungen, die das betrachtete Verfahren besonders gut klassifizieren kann, fällt die Klassifizierungsgenauigkeit höher aus als in anderen Fällen. Die Berechnung der Klassifizierungsgenauigkeit macht dann Sinn, wenn die Testdaten in ihrer Zusammensetzung dem Verkehr in einem realen Netz entsprechen. Im Idealfall sollten die Testdaten also Verkehr enthalten, der an einem Punkt in einem realen Netz gemessen wurden.

Da in unserem Fall die Testdaten aus verschiedenen Messungen zusammengesetzt und damit nicht repräsentativ für realen Netzverkehr sind, verzichten wir auf die Angabe der Klassifizierungsgenauigkeit. Stattdessen berechnen wir den Recall-Mittelwert über alle Anwendungen, um verschiedene Verfahren und Parametrisierungen vergleichen zu können. Dieser Wert entspricht der Klassifizierungsgenauigkeit in einem Netz, in dem jede Anwendung mit gleich vielen Verbindungen vertreten ist.

C. Klassifizierung anhand von TCP-Flags

In diesem Unterabschnitt stellen wir die Ergebnisse für den Fall vor, dass die TCP-Flags und die Flussrichtung der Pakete die Zustände im Markov-Modell definieren. Dadurch ergibt sich ein Markov-Modell mit 128 Zuständen, 64 für jede Richtung. Tabelle II zeigt die Ergebnisse für verschiedene Maximalanzahlen von Paketen, die je Verbindung berücksichtigt werden. Die besten Recall- und Precision-Werte sind für jede Anwendung fett gedruckt. Um das Verfahren für die Online-Klassifizierung einsetzen zu können, sollten pro Verbindung möglichst wenige Pakete untersucht werden.

In Tabelle II ist zu sehen, dass bereits gute Ergebnisse erzielt werden können, wenn 20 Pakete pro Verbindung untersucht werden. Die Klassifikationsergebnisse werden im Allgemeinen schlechter, wenn die Anzahl der untersuchten Pakete deutlich größer als 30 wird. Am schlechtesten sind die Ergebnisse, wenn sämtliche Pakete untersucht werden. Diese Beobachtung legt nahe, dass die ersten Pakete einer Verbindung charakteristische Übergängen zwischen TCP-Flag-Kombinatinen aufweisen, wodurch sich die Anwendungen recht gut unterscheiden lassen. Betrachtet man komplette Verbindungen, gehen auch spätere Übergänge, die weniger charakteristisch sind und nur in langen Verbindungen mit vielen Paketen auftreten, in die Markov-Modelle ein. Dadurch verwischen sich die Unterschiede zwischen den verschiedenen Anwendungen, was sich durch sehr ähnliche Werte in den Übergangsmatrizen der verschiedenen Anwendungen bemerkbar macht.

Die selben Untersuchungen wurden auch für Markov-Modelle durchgeführt, bei denen die Flussrichtung der Pakete unberücksichtigt blieb, d.h. für Markov-Modelle mit 64 Zuständen für die unterschiedlichen TCP-Flag-Kombinationen. Unabhängig davon, ob alle Pakete einer Verbindung oder nur die Pakete einer Flussrichtung (von Client zu Server oder umgekehrt) bei der Modellbildung und Klassifizierung betrachtet wurden, ergaben sich schlechtere Klassifikationsergebnisse als in Tabelle II. Die Flussrichtung stellte sich also in Zusammenhang mit den TCP-Flags als wichtiges Paketmerkmal heraus.

D. Klassifizierung anhand von Paketlängen

Nun gehen wir auf die Ergebnisse ein, die sich für Markov-Modelle ergeben, deren Zustände sich aus den Paketgrößen und der Flussrichtung ableiten. Wir bilden hierfür die Paketgrößen auf 16 Intervalle ab, die sich an der Nutzdatenlänge orientieren, d.h. der Paketlänge abzüglich des TCP/IP-Headers. Der erste Zustand repräsentiert Pakete ohne Nutzdaten (Nutzdatenlänge ist Null). Die folgenden 14 Zustände bilden Intervalle mit einer Länge von je 100 Byte ab: $[1:100], [101:200], \ldots, [1301:1400]$. Der letzte Zustand steht für Pakete mit Nutzdatenlängen ab 1401 Byte. Diese Intervallaufteilung ist willkürlich, bietet aber einen Kompromiss zwischen einer feingranularen Unterscheidung von Paketgrößen und einer überschaubaren Zustandsanzahl. Zusammen mit der Flussrichtung ergeben sich somit 32 verschiedene Zustände.

Tabelle III liefert die Ergebnisse für verschiedene Maximalanzahlen von Paketen, die je Verbindung berücksichtigt

Tabelle II: Klassifikationsergebnis bei Berücksichtigung von TCP-Flags und Flussrichtung

Maximale Paketanzahl	(Recall, Precision) in Abhängigkeit von der Anzahl untersuchter Pakete					
	10	15	20	30	100	alle
eDonkey	41,25%, **87,85%**	51,90%, 65,63%	70,53%, 73,90%	**80,42%**, 76,22%	70,15%, 69,89%	17,87%, 59,49%
BitTorrent	65,80%, 69,57%	**83,77%**, 91,06%	81,39%, **93,53%**	80,95%, 92,35%	78,57%, 87,47%	7,79%, 31,58%
Gnutella	90,38%, 77,69%	88,46%, 80,23%	**94,23%**, **82,58%**	93,59%, 74,49%	92,63%, 69,14%	83,97%, 38,47%
HTTP	80,95%, 97,62%	**85,28%**, **97,89%**	84,63%, 95,98%	85,28%, 91,02%	80,29%, 80,39%	59,13%, 79,51%
IMAP	75,63%, 63,20%	71,76%, 68,65%	86,05%, 79,63%	**89,24%**, 92,99%	73,28%, **95,61%**	80,00%, 52,48%
SMTP	**96,99%**, 59,16%	86,14%, 66,67%	79,13%, 78,87%	88,98%, 90,49%	94,32%, **92,93%**	89,15%, 66,92%
SSH	62,33%, **96,31%**	84,08%, 87,33%	93,37%, 82,24%	**93,63%**, 89,82%	86,47%, 79,32%	27,32%, 28,30%
YouTube	92,93%, 92,63%	96,78%, **95,86%**	95,18%, 92,21%	95,50%, 95,19%	**99,36%**, 93,35%	97,43%, 91,27%
⌀-Recall	75,78%	81,02%	85,56%	**88,44%**	84,38%	57,83%

Tabelle III: Klassifikationsergebnis bei Berücksichtigung von Nutzdatenlänge (16 Intervalle) und Flussrichtung

Maximale Paketanzahl	(Recall, Precision) in Abhängigkeit von der Anzahl untersuchter Pakete					
	10	15	20	30	100	alle
eDonkey	80,23%, **95,91%**	81,94%, 80,71%	81,18%, 94,89%	**85,55%**, 91,65%	45,06%, 86,50%	9,70%, 40,48%
BitTorrent	**96,54%**, 95,71%	95,24%, 95,24%	95,02%, 93,21%	93,94%, **97,75%**	89,83%, 95,84%	3,68%, 3,83%
Gnutella	82,69%, **92,47%**	70,51%, 85,27%	82,05%, 87,67%	**84,94%**, 88,93%	82,37%, 75,81%	81,09%, 25,66%
HTTP	87,52%, 95,69%	86,20%, 95,77%	**89,09%**, 96,72%	87,91%, **97,38%**	83,71%, 94,65%	32,06%, 65,42%
IMAP	90,92%, 90,47%	**95,97%**, 94,85%	95,80%, 94,37%	90,42%, **95,73%**	75,97%, 79,02%	65,71%, 58,62%
SMTP	**95,16%**, 85,33%	77,63%, 85,79%	86,64%, 91,70%	86,64%, **92,18%**	75,63%, 90,06%	65,28%, 91,78%
SSH	**98,67%**, **99,20%**	97,61%, 92,93%	98,41%, 93,45%	97,61%, 94,85%	93,37%, 62,19%	91,51%, 56,01%
YouTube	95,18%, **100%**	95,18%, 97,05%	94,53%, 98,33%	93,89%, 98,65%	**95,82%**, 93,42%	38,26%, 86,23%
⌀-Recall	**90,86%**	87,53%	90,34%	90,11%	80,22%	48,41%

werden. Ähnlich wie in Unterabschnitt IV-C können wir feststellen, dass schon mit wenigen Paketen pro Verbindung ein gutes Klassifikationsergebnis erreicht werden kann. In den meisten Fällen sind die Werte für Recall und Precision besser als bei der Verwendung der TCP-Flags. Dies deutet darauf hin, dass die Übergänge zwischen verschiedenen Paketgrößen bzw. Nutzdatenlängen charakteristischer für eine Anwendung sind als die Übergänge zwischen den gesetzten TCP-Flag-Kombinationen. Bereits bei zehn bis 15 Paketen pro Verbindung ist eine hohe Klassifizierungsgenauigkeit zu beobachten. Auch hier werden die Klassifikationsergebnisse schlechter, wenn mehr als 30 Pakete je Verbindung betrachtet werden, wobei die Paketanzahl mit dem größten Recall- oder Precision-Wert von Anwendung zu Anwendung variiert. Diese Beobachtung deckt sich mit den Ergebnissen von Bernaille, der mit seinem Verfahren die besten Ergebnisse erzielte, wenn er es auf die ersten vier Datenpakete beschränkte [1]. Dass bei uns der größte mittlere Recall-Wert bei zehn bis 15 Paketen erreicht wird, hängt damit zusammen, dass wir im Gegensatz zu Bernaille sämtliche Pakete einer Verbindung betrachten, also auch diejenigen ohne Nutzdaten.

Im Folgenden untersuchen wir, welchen Einfluss die Berücksichtigung der Flussrichtung der Pakete auf das Klassifikationsergebnis hat. Bisher hatten wir die Flussrichtung bei der Abbildung auf die Zustände mitberücksichtigt. Dies führte zu einer Verdoppelung der Zustandsanzahl, weil jedes Paketgrößenintervall für beide Richtungen getrennt auf je einen Zustand abgebildet wurde. Wenn man diese Verdoppelung vermeiden möchte, kann man entweder die Flussrichtung gänzlich unberücksichtigt lassen oder sich auf eine der beiden Flussrichtungen beschränken.

Die Tabellen IV und V zeigen die Klassifikationsergebnisse für alle vier Möglichkeiten im Vergleich, wobei jeweils die ersten 10 bzw. 15 Pakete einer Verbindung oder Flussrichtung zur Klassifizierung verwendet wurden. Die erste Spalte zeigt die Ergebnisse für Recall und Precision, wenn die Flussrichtung unberücksichtigt bleibt. In der zweiten und dritten Spalte werden bei der Modellbildung und Klassifizierung nur Pakete von Client zu Server („C→S“) bzw. Server zu Client („S→C“) verwendet. Die letzte Spalte wiederholt zum Vergleich die entsprechenden Ergebnisse aus Tabelle III (Flussrichtung „zustandskodiert“). Man sieht, dass die Berücksichtigung der Flussrichtung in der Zustandskodierung für die meisten Anwendungen das beste Ergebnis liefert.

Interessant ist, dass speziell die Erkennung der Peer-to-Peer-Protokolle (eDonkey, BitTorrent, Gnutella) sehr stark auf Richtungsinformationen angewiesen ist, wobei sich kleine Unterschiede ergeben. Gnutella kann nur sehr schlecht erkannt werden, wenn die Flussrichtung unberücksichtigt bleibt oder nur eine Flussrichtung untersucht wird. BitTorrent liefert die schlechtesten Recall-Werte, wenn nur Pakete einer Richtung betrachtet werden. eDonkey erreicht zwar die besten Recall-Werte, wenn nur der Verkehr von Server zu Client untersucht wird, allerdings bei einer sehr niedrigen Precision von 52 bzw. 44 Prozent.

Mit Ausnahme von IMAP und SSH, wo auch anhand der Pakete von Server zu Client gute Ergebnisse erzielt werden, verbessern sich bei den Client-Server-Protokollen die Ergebnisse wesentlich, wenn beide Flussrichtungen betrachtet werden. In den wenigen Fällen, in denen die Analyse ohne Berücksichtigung der Flussrichtung die besten Ergebnisse liefert, ist der Unterschied zur Variante, bei der die Richtung in den Zuständen abgebildet ist, recht gering.

Wichtig bei der Interpretation der Zahlenwerte ist, dass

Tabelle IV: Einfluss der Flussrichtung auf Recall und Precision (10 untersuchte Pakete pro Verbindung)

Flussrichtung	Zustände abgeleitet aus Nutzdatenlänge			
	unberücksichtigt	nur C→S	nur S→C	zustandskodiert
eDonkey	75,10%, 76,40%	67,49%, 43,35%	**81,37%**, 52,13%	80,23%, **95,91%**
BitTorrent	84,20%, 89,22%	31,82%, 60,74%	34,85%, 78,16%	**96,54%**, **95,71%**
Gnutella	54,17%, 70,71%	66,99%, 74,64%	46,47%, 65,32%	**82,69%**, **92,47%**
HTTP	**90,41%**, 93,61%	53,48%, 73,33%	70,96%, 90,00%	87,52%, **95,69%**
IMAP	88,57%, 91,65%	90,25%, 68,58%	**96,81%**, **90,57%**	90,92%, 90,47%
SMTP	94,99%, 81,29%	61,44%, 81,96%	75,46%, 76,09%	**95,16%**, **85,33%**
SSH	98,67%, 98,67%	71,35%, 75,99%	98,41%, 97,63%	**98,67%**, **99,20%**
YouTube	91,96%, 95,97%	80,06%, 58,59%	76,85%, 61,76%	**95,18%**, **100%**
∅-Recall	84,75%	65,36%	72,64%	**90,86%**

Tabelle V: Einfluss der Flussrichtung auf Recall und Precision (15 untersuchte Pakete pro Verbindung)

Flussrichtung	Zustände abgeleitet aus Nutzdatenlänge			
	unberücksichtigt	nur C→S	nur S→C	zustandskodiert
eDonkey	45,44%, 77,10%	65,78%, 37,24%	**85,55%**, 44,60%	81,94%, **80,71%**
BitTorrent	88,74%, 70,93%	31,39%, 62,77%	34,42%, 71,30%	**95,24%**, **95,24%**
Gnutella	47,76%, 74,13%	46,47%, 70,39%	41,67%, 56,52%	**70,51%**, **85,27%**
HTTP	**90,67%**, **95,97%**	48,09%, 75,93%	64,52%, 89,60%	86,20%, 95,77%
IMAP	93,28%, 77,08%	76,47%, 71,77%	86,55%, 83,33%	**95,97%**, **94,85%**
SMTP	77,30%, 71,78%	60,27%, 59,97%	58,26%, 84,50%	**77,63%**, **85,79%**
SSH	**98,94%**, 95,64%	71,35%, 70,60%	97,35%, **98,39%**	97,61%, 92,93%
YouTube	**96,14%**, 94,92%	78,46%, 57,55%	77,17%, 59,70%	95,18%, **97,05%**
∅-Recall	79,78%	59,78%	68,18%	**87,53%**

die Ergebnisse für die unterschiedlichen Anwendungen nicht isoliert betrachtet werden dürfen, sondern in engem Zusammenhang stehen. So ist beispielsweise ein guter Recall-Wert für eine Anwendung nicht nur darauf zurückzuführen, dass das zugehörige Markov-Modell die Anwendung gut charakterisiert, sondern auch darauf, dass die Markov-Modelle der übrigen Anwendungen weit weniger gut passen.

E. Klassifizierung anhand von TCP-Flags und Paketlängen

Im Folgenden untersuchen wir die Kombination von Nutzdatenlängen und TCP-Flags als Basis der Verkehrsklassifizierung mit Markov-Modellen. Betrachtet man die TCP-Flags der Pakete einer Verbindung (siehe Abbildung 1), stellt man fest, dass für die Unterscheidung verschiedener Anwendungen nur wenige TCP-Flag-Kombinationen von Interesse sind. So treten SYN- und SYN/ACK-Pakete nur zu Beginn einer Verbindung auf, das FIN- und RST-Flag nur am Verbindungsende. Dadurch tragen diese Flags kaum zur Unterscheidbarkeit verschiedener Anwendungen bei. Das URG-Flag hat so gut wie keine Bedeutung, weil es in der Praxis – wenn überhaupt – nur äußerst selten verwendet wird. In unseren Test- und Trainingsdaten war es in keinem einzigen Paket gesetzt. Am aussagekräftigsten ist das PUSH-Flag, welches immer in Kombination mit dem ACK-Flag auftritt. Für die kombinierte Betrachtung von TCP-Flags und Nutzdatenlängen bietet es sich daher an, nur das PUSH-Flag zu berücksichtigen, um die Zustandsanzahl gering zu halten.

Eine weitere Frage betrifft die Anzahl der Intervalle, in die man die Nutzdatenlängen einteilt. In einem ersten Ansatz behalten wir die Aufteilung in 16 Intervalle bei, die bereits im vorhergehenden Unterabschnitt verwendet wurde. Insgesamt führt dies zu 64 Zuständen, da wir beide Flussrichtungen getrennt betrachten und zusätzlich das Vorhandensein des PUSH-Flags berücksichtigen. In Tabelle VI sind die Klassifikationsergebnisse für diese Zustandsdefinitionen dargestellt. Der Vergleich mit den Tabellen II und III zeigt, dass die Kombination beider Paketeigenschaften etwas bessere Ergebnisse liefert als die alleinige Betrachtung von Nutzdatenlänge oder TCP-Flags. Wieder reichen wenige Pakete pro Verbindung aus, um gute Klassifikationsergebnisse zu erzielen.

In weiteren Untersuchungen verringerten wir die Zustandsanzahl, indem wir die Nutzdatenlängen in weniger Intervalle aufteilten. Als günstig erwies sich dabei die Aufteilung in vier Intervalle: $[0:0], [1:299], [300:(\mathrm{MSS}-1)], [\mathrm{MSS}]$. MSS bezeichnet hierbei die *Maximum Segment Size* der TCP-Verbindung, die der maximal möglichen Nutzdatenlänge entspricht. Durch diese Intervallaufteilung ergeben sich insgesamt nur 16 anstelle der vorherigen 64 Zustände, was einen geringeren Speicherbedarf für die Matrix der Übergangswahrscheinlichkeiten zur Folge hat. Tabelle VII listet die zugehörigen Klassifikationsergebnisse auf. Es ist zu beobachten, dass sich die Ergebnisse gegenüber der Verwendung von 64 Zuständen nicht verschlechtert haben. Für einige Anwendungen, wie zum Beispiel bei Gnutella, HTTP, SMTP und YouTube, konnte die Genauigkeit der Klassifikation sogar gesteigert werden. Mit Ausnahme von eDonkey-Verkehr liegen die Recall-Werte bei maximal zehn untersuchten Paketen über 90 Prozent, für die meisten Anwendungen sogar über 95 Prozent, was ein sehr gutes Ergebnis ist. Der durchschnittliche Recall-Wert liegt bei 94,3 Prozent.

F. Vergleich mit Verfahren von Bernaille

Im Folgenden vergleichen wir unseren Ansatz mit dem Verfahren von Bernaille [1], das als Matlab-Quellcode von einer

Tabelle VI: Klassifikationsergebnis bei Berücksichtigung von Nutzdatenlänge (16 Intervalle), PUSH-Flag und Flussrichtung

Maximale Paketanzahl	(Recall, Precision) in Abhängigkeit von der Anzahl untersuchter Pakete				
	10	15	20	30	100
eDonkey	83,27%, 94,60%	80,42%, 94,42%	85,17%, 97,18%	**86,31%, 97,63%**	70,34%, 96,86%
BitTorrent	**95,24%**, 97,56%	93,94%, 97,75%	92,21%, 98,38%	92,21%, **99,07%**	90,26%, 98,58%
Gnutella	**90,71%, 95,93%**	86,54%, 95,74%	79,49%, 92,54%	79,17%, 94,64%	77,88%, 91,01%
HTTP	**84,49%**, 98,02%	83,18%, 98,91%	84,23%, 98,46%	82,79%, 98,13%	80,03%, **99,02%**
IMAP	91,26%, 93,46%	**96,81%**, 93,51%	95,80%, 92,53%	88,74%, **97,42%**	81,18%, 90,11%
SMTP	**96,83%**, 89,64%	90,48%, 93,93%	86,48%, **96,10%**	90,32%, 94,09%	84,97%, 89,14%
SSH	**97,61%, 100%**	96,55%, 97,85%	97,35%, 97,35%	94,69%, 97,81%	90,45%, 87,66%
YouTube	95,18%, **100%**	**96,14%**, 96,76%	93,89%, 98,32%	92,93%, 97,97%	96,14%, 95,22%
⌀-Recall	**91,82%**	90,50%	89,32%	88,39%	83,90%

Tabelle VII: Klassifikationsergebnis bei Berücksichtigung von Nutzdatenlänge (4 Intervalle), PUSH-Flag und Flussrichtung

Maximale Paketanzahl	(Recall, Precision) in Abhängigkeit von der Anzahl untersuchter Pakete				
	10	15	20	30	100
eDonkey	82,32%, 92,92%	81,56%, 92,26%	84,98%, 92,36%	**92,78%, 93,31%**	73,19%, 91,89%
BitTorrent	**94,59%**, 97,33%	94,37%, 96,25%	89,18%, 97,40%	88,53%, **98,08%**	86,15%, 97,07%
Gnutella	**95,19%**, 93,69%	90,38%, **94,00%**	90,38%, 92,46%	91,35%, 92,53%	89,74%, 83,09%
HTTP	**96,58%**, 99,06%	93,96%, **99,58%**	95,40%, 99,05%	93,69%, 99,03%	90,80%, 99,00%
IMAP	92,44%, 92,75%	**96,30%**, 92,42%	95,97%, 90,92%	95,13%, **96,42%**	91,26%, 89,90%
SMTP	**97,16%**, 88,05%	96,33%, 92,77%	94,32%, **95,28%**	92,65%, 93,91%	86,98%, 86,69%
SSH	97,35%, **100%**	**98,14%**, 96,86%	98,14%, 92,04%	96,82%, 87,53%	95,23%, 75,90%
YouTube	**99,04%**, 98,40%	98,71%, **99,03%**	96,78%, 98,37%	97,11%, 97,11%	98,39%, 92,73%
⌀-Recall	**94,33%**	93,72%	93,14%	93,51%	89,97%

Projektwebseite heruntergeladen werden kann [12]. Wie in Unterabschnitt IV-A erwähnt, haben wir die Trainings- und Testdaten so gewählt, dass jeweils mindestens vier Datenpakete pro Verbindung enthalten sind, um Bernailles Verfahren anwenden zu können. Unser Verfahren arbeitet aber grundsätzlich auch mit Verbindungen, die weniger Datenpakete enthalten.

Die vorliegende Implementierung von Bernaille setzt GMM als Cluster-Algorithmus ein, um Verbindungen mit ähnlichen Nutzdatenlängen in den ersten Datenpaketen zu gruppieren. Dabei wird zu Beginn eine zufällige Cluster-Aufteilung initialisiert. Abhängig von dieser Initialisierung ergeben sich unterschiedliche Cluster und damit auch unterschiedliche Klassifikationsergebnisse. Weiterhin muss der Cluster-Algorithmus mit der Anzahl der zu bestimmenden Cluster parametrisiert werden. Auch von diesem Parameter hängt das Klassifikationsergebnis ab.

Bernaille empfiehlt in der Dokumentation seines Matlab-Quellcodes, die Nutzdatenlängen der ersten vier Datenpakete in 40 Cluster aufzuteilen. Kleine Cluster, denen weniger als drei Verbindungen aus den Testdaten zugeordnet werden, werden durch seinen Algorithmus automatisch entfernt. Mit diesen empfohlenen Parametern haben wir fünf Durchläufe mit unterschiedlichen Initialisierungen durchgeführt.

In Tabelle VIII sind die Klassifikationssergebnisse aufgelistet, die sich aufgrund der Cluster-Zuordnungen der Verbindungen in den Trainingsdaten ergeben. Wir verwenden also nicht die Variante des Verfahrens, die Portnummern mitberücksichtigt, da auch unser Verfahren Portnummern außer Acht lässt. Wie man aus der Tabelle entnehmen kann, fallen die Ergebnisse je nach Initialisierung sehr unterschiedlich aus. Häufig finden sich einzelne Anwendungen, die schlecht klassifiziert werden, wie zum Beispiel YouTube im ersten Versuch, wo der Recall-Wert nur 49 Prozent beträgt. Im Vergleich zum dritten Versuch, wo 89 Prozent für diese Anwendung erreicht werden, ist dies ein sehr großer Unterschied. Ein ähnliches Beispiel liefert Gnutella, das im ersten Versuch mit einem Recall-Wert von über 99 Prozent sehr gut abschneidet, im letzten Versuch aber nur 75 Prozent erreicht.

Der beste durchschnittliche Recall-Wert liegt bei knapp 92 Prozent, während wir mit unserem Verfahren über 94 Prozent erreichen konnten (siehe Tabelle VII). Auch bei nicht optimal gewählter maximaler Paketanzahl lagen bei uns die mittleren Recall-Werte meist deutlich über 90 Prozent. Ein wesentlicher Vorteil unseres Verfahrens ist, dass die Bildung der Markov-Modelle nicht von der Wahl geeigneter Startwerte abhängt sondern für gegebene Trainingsdaten deterministische Ergebnisse liefert. Auch sind bei unserem Verfahren keine größeren Ausreißer bei einzelnen Anwendungen zu beobachten.

Bernaille beschreibt, wie mit Hilfe des normalisierten mittleren Transinformationsgehalts (engl. *Normalized Mutual Information*) die besten Werte für die Cluster-Anzahl und die Anzahl zu untersuchender Datenpakete bestimmt werden können [1]. Im Matlab-Quellcode von Bernaille ist ein entsprechender Kalibrierungsalgorithmus enthalten, der verschiedene Parameterkombinationen durchprobiert und für die Kombination mit dem größten normalisierten mittleren Transinformationsgehalt das Cluster-Ergebnis zurückgibt. Dabei wird der Cluster-Algorithmus auch hier mit einer zufälligen Cluster-Aufteilung initialisiert, wodurch sich wieder in jedem Durchlauf ein anderes Ergebnis ergibt. In Tabelle IX sind die Kalibrierungs- und Klassifikationsergebnisse für sechs Durchläufe aufgeführt. Als mögliche Parameter waren drei oder vier Datenpakete erlaubt sowie eine anfängliche Cluster-Anzahl von 30, 35 oder 40. Abweichende Cluster-Zahlen

Tabelle VIII: Klassifizierungsergebnisse für Bernaille mit 4 Datenpaketen und maximal 40 Clustern

	Berücksichtigung von 4 Datenpaketen, 40 Cluster (abzüglich Cluster mit weniger als 3 Verbindungen)					
	1. Versuch	2. Versuch	3. Versuch	4. Versuch	5. Versuch	6. Versuch
eDonkey	99,43%, 94,23%	99,05%, 93,04%	99,81%, 88,83%	89,35%, 99,58%	96,20%, 96,20%	90,11%, 92,76%
BitTorrent	92,86%, 99,31%	96,75%, 99,55%	95,24%, 81,18%	90,48%, 100%	93,94%, 97,09%	88,10%, 90,44%
Gnutella	99,68%, 64,26%	88,14%, 100%	98,72%, 73,86%	95,51%, 71,29%	98,40%, 71,23%	75,32%, 78,07%
HTTP	71,22%, 78,89%	83,71%, 83,93%	73,46%, 93,95%	78,45%, 91,85%	80,16%, 90,77%	89,09%, 87,82%
IMAP	97,31%, 96,98%	89,75%, 100%	92,10%, 100%	91,93%, 100%	96,81%, 88,07%	91,43%, 99,82%
SMTP	92,65%, 100%	94,99%, 94,36%	95,83%, 100%	97,50%, 94,50%	97,66%, 100%	98,00%, 99,83%
SSH	97,61%, 100%	97,88%, 100%	71,88%, 98,55%	96,82%, 98,92%	96,82%, 100%	97,08%, 98,92%
YouTube	49,20%, 57,74%	84,89%, 67,01%	89,07%, 69,08%	86,17%, 59,42%	69,77%, 82,51%	78,14%, 59,85%
⌀-Recall	87,49%	**91,89%**	89,51%	90,81%	91,22%	88,41%

Tabelle IX: Kalibrierung der Parameter mit Hilfe des normalisierten mittleren Transinformationsgehaltes

	Kalibrierung der Datenpaketanzahl (3 oder 4) und der Cluster-Anzahl (Startwerte 30, 35 oder 40)					
	1. Versuch	2. Versuch	3. Versuch	4. Versuch	5. Versuch	6. Versuch
	4 Pakete 29 Cluster	3 Pakete 37 Cluster	3 Pakete 29 Cluster	3 Pakete 33 Cluster	3 Pakete 28 Cluster	3 Pakete 28 Cluster
eDonkey	92,78%, 96,44%	91,06%, 96,96%	94,11%, 80,36%	91,25%, 96,00%	96,96%, 92,06%	92,59%, 92,94%
BitTorrent	99,13%, 86,42%	98,27%, 93,22%	82,03%, 100%	97,40%, 94,54%	97,40%, 98,25%	91,56%, 99,06%
Gnutella	79,49%, 66,49%	86,54%, 100%	93,27%, 85,34%	96,79%, 80,32%	98,40%, 64,77%	86,86%, 85,76%
HTTP	75,56%, 90,55%	80,29%, 88,17%	71,09%, 95,75%	80,03%, 91,30%	72,93%, 93,91%	41,13%, 72,45%
IMAP	96,81%, 91,57%	95,80%, 96,61%	92,61%, 85,16%	95,97%, 92,39%	89,92%, 98,89%	86,72%, 99,61%
SMTP	97,66%, 100%	81,30%, 100%	98,50%, 100%	94,32%, 100%	98,83%, 94,42%	98,00%, 94,37%
SSH	96,02%, 100%	99,73%, 78,01%	95,49%, 100%	97,35%, 98,39%	97,08%, 100%	98,67%, 100%
YouTube	78,14%, 75,23%	90,35%, 63,86%	83,92%, 58,65%	84,57%, 71,47%	83,60%, 78,31%	88,75%, 37,70%
⌀-Recall	89,44%	90,41%	88,87%	**92,21%**	91,89%	85,53%

ergeben sich durch das automatische Entfernen kleiner Cluster, denen weniger als drei Verbindungen aus den Testdaten zugeordnet werden.

Der beste mittlere Recall-Wert von knapp über 92 Prozent ergibt sich im vierten Versuch, wobei hier die Recall-Werte für HTTP und YouTube bei vergleichsweise schlechten 80 und 85 Prozent liegen. Auffällig an den Ergebnissen ist, dass sich durch die Kalibrierung keine deutliche Verbesserung der Klassifikationsergebnisse einstellt. Die Anzahl der Cluster wie auch die Anzahl der Datenpakete scheint also weit weniger Einfluss auf das Ergebnis zu haben als die zufällige Initialisierung des Cluster-Algorithmus.

G. Klassifizierungsaufwand

Zur Klassifizierung einer Verbindung wird für jedes Markov-Modell der $LogMAP$-Wert nach Gleichung (4) berechnet. Die Berechnung eines $LogMAP$-Wertes erfordert $(N-1)$ Additionen, wobei N das Minimum aus der Anzahl der Pakete in der Verbindung und der vorgegebenen Maximalanzahl berücksichtigter Pakete ist. Zur Klassifizierung einer Verbindung sind bei K Anwendungen also maximal $K(N-1)$ Additionen und $(K-1)$ Vergleiche notwendig. Unser Verfahren zeichnet sich somit durch eine sehr geringe Komplexität aus und eignet sich daher sehr gut für die Online-Verkehrsklassifizierung.

Beim Verfahren von Bernaille muss dagegen für jeden Cluster die Wahrscheinlichkeit für die Zugehörigkeit einer gegebenen Verbindung berechnet werden, um die Verbindung danach dem wahrscheinlichsten Cluster zuordnen zu können. Im Fall von GMM werden die Zugehörigkeitswahrscheinlichkeiten unter der Normalverteilungsannahme ermittelt, was eine vergleichsweise aufwendige Berechnung erfordert. Damit ist die Klassifizierung einer Verbindung um einiges aufwendiger als bei unserem Verfahren.

Für ein gegebenes Verkehrsaufkommen nimmt der Klassifizierungsaufwand bei beiden Verfahren linear mit der Anzahl der Verbindungen zu.

V. Zusammenfassung und Ausblick

In der vorliegenden Arbeit haben wir ein Verfahren zur Verkehrsklassifizierung vorgestellt, das die charakteristische Abfolge von TCP-Flags und Paketlängen in den TCP-Verbindungen einer Anwendung in Markov-Modellen modelliert. Mit Hilfe von Trainingsdaten bestimmen wir ein Markov-Modell für jede Anwendung und verwenden diese Modelle zur Klassifikation neuer TCP-Verbindungen. Hierzu genügt es, das Modell mit der maximalen A-posteriori-Wahrscheinlichkeit zu bestimmen. Anhand von echten Verkehrsaufnahmen konnten wir zeigen, dass sich die Markov-Modelle verschiedener Anwendungen insbesondere bei der kombinierten Betrachtung von TCP-Flags und Paketlängen ausreichend stark unterscheiden, um verschiedene Anwendungen zuverlässig voneinander abgrenzen und somit eine hohe Klassifikationsgenauigkeit erreichen zu können. Unsere Auswertung ergab zudem, dass die Untersuchung der ersten zehn Pakete einer TCP-Verbindung ausreicht. Das vorgestellte Verfahren eignet sich daher hervorragend für die Online-Klassifizierung von Verkehrsströmen.

In einem direkten Vergleich mit dem viel zitierten Verfahren von Bernaille erzielte unser Ansatz bessere Klassifikationsergebnisse. Ein wesentlicher Vorteil ist zudem, dass die Trainingsphase bei unserem Ansatz ein deterministisches Modell ergibt, während der von Bernaille verwendete Cluster-Algorithmus auf zufällig gewählte Initialisierungswerte ange-

wiesen ist, die das Ergebnis unserer Erfahrung nach recht stark beeinflussen.

Die für die Evaluierung verwendeten Verkehrsdaten wurden an den Servern unserer Arbeitsgruppe sowie an Rechnern, auf denen Peer-to-Peer-Anwendungen liefen, aufgezeichnet. Wir haben vor, diese Untersuchungen mit weiteren Verkehrsdaten zu wiederholen, die aus größeren Netzen stammen und eine vielfältigere Verkehrszusammensetzung beinhalten. In diesem Zusammenhang ist dann auch zu untersuchen, wie mit Verkehrsströmen unbekannter Anwendungen umgegangen werden kann. Bisher klassifizieren wir eine Verbindung nur dann als einer unbekannten Anwendung zugehörig, wenn ein Zustandsübergang beobachtet wird, der in den Trainingsdaten nicht auftrat (Übergangswahrscheinlichkeit ist Null). Falls die betrachteten Zustandsübergänge in der TCP-Verbindung der unbekannten Anwendung auch in einem der Markov-Modell existieren, wird die Verbindung fälschlicherweise einer bekannten Anwendung zugeordnet. Eine Möglichkeit, solche Fehler zu vermeiden, besteht darin, einen unteren Schwellwert für die maximale A-posteriori-Wahrscheinlichkeit einzuführen und eine TCP-Verbindung als Verkehr einer unbekannten Anwendung anzusehen, wenn dieser Schwellwert unterschritten wird.

Schließlich möchten wir versuchen, unser Verfahren weiter zu verbessern, indem wir die Position eines Pakets innerhalb der TCP-Verbindung in die Markov-Modelle mit einfließen lassen. Bei vielen Anwendungen ist die Abfolge der Paketlängen am Anfang einer TCP-Verbindung sehr charakteristisch – eine Eigenschaft, auf der das Verfahren von Bernaille im Wesentlichen beruht. In den Markov-Modellen könnte dies durch Berechnung positionsabhängiger Übergangswahrscheinlichkeiten stärker berücksichtigt werden.

DANKSAGUNG

Wir danken der Deutschen Forschungsgemeinschaft (DFG) für die Förderung des LUPUS-Projekts (DFG-Geschäftszeichen: CA 595/1-1), in dessen Rahmen die vorgestellte Forschungsarbeit durchgeführt wurde.

LITERATUR

[1] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proc. of ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, Lisboa, Portugal, Dec. 2006.

[2] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proc. of Passive and Active Network Measurement Workshop (PAM)*, Boston, USA, Mar. 2005.

[3] H.-c. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proc. of ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, Madrid, Spain, Dec. 2008.

[4] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[5] L. Bernaille, R. Teixeira, I. Akodjenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, Apr. 2006.

[6] L. Bernaille, A. Soule, I. Akodjenou, and K. Salamatian, "Blind application recognition through behavioral classification," CNRS LIP6, Paris, France, Tech. Rep., 2005.

[7] P. Smyth, "Clustering sequences with hidden markov models," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, Denver, USA, Dec. 1996, pp. 648–654.

[8] F. Porikli, "Trajectory distance metric using hidden markov model based representation," in *Proc. of European Conference on Computer Vision (ECCV)*, Prague, Czech Republic, May 2004.

[9] C. Wright, F. Monrose, and G. Masson, "HMM profiles for network traffic classification (extended abstract)," in *Proc. of Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*, Fairfax, VA, USA, Oct. 2004, pp. 9–15.

[10] A. Dainotti, W. d. Donato, A. Pescapè, and P. S. Rossi, "Classification of network traffic via packet-level hidden markov models," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, New Orleans, USA, Nov. 2008.

[11] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo, "Stochastic protocol modeling for anomaly based network intrusion detection," in *Proc. of IEEE International Workshop on Information Assurance (IWIA)*, Mar. 2003.

[12] L. Bernaille, "Homepage of early application identification," http://www-rp.lip6.fr/ teixeira/bernaill/earlyclassif.html, 2009.