Schlussbericht

Projektname SIAN

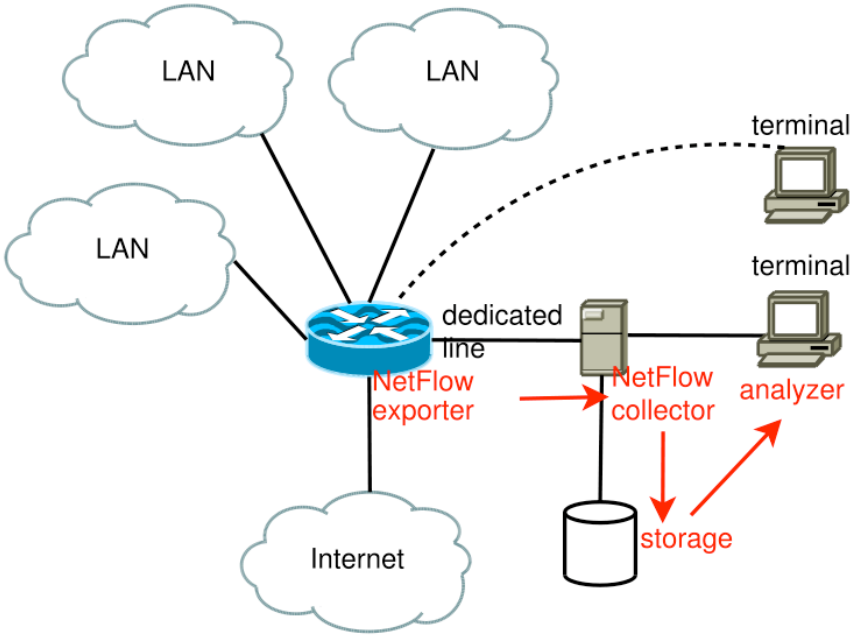
Software zur Identifikation von Applikationsklassen in NetFlow

Typ der Arbeit	Semesterarbeit
Studiengang	Informatik
Semester	FS 2009
Betreuer	HSR, Eduard Glatz
Auftraggeber	ETH Zürich, Bernhard Tellenbach

Autoren	**SEAK**	Akyol Selim
	EDBE	Berisha Edon

Ausgabedatum

Verteiler Betreuer: HSR, Eduard Glatz
Auftraggeber: ETH Zürich, Bernhard Tellenbach



0. Inhalt

1. Einführung

1.1. Zweck

Dieses Dokument beinhaltet die komplette Dokumentation über die Semesterarbeit des Projektes SIAN. Es werden die technischen Aspekte und die persönlichen Berichte in diesem Dokument zusammengefasst.

1.2. Gültigkeitsbereich

Dieses Dokument ist für die gesamte Projektzeit gültig. Änderungen bleiben vorbehalten.

1.3. Definitionen und Abkürzungen

Ein Glossar mit den wichtigsten Begriffen, die für das Verständnis des Projektes notwendig sind, wurde in ein separates Dokument erstellt.

Siehe Kapitel 6 „Glossar“.

2. Aufgabenstellung

3. Abstract

Abteilung	**Informatik**
Name der Studierenden	**Selim Akyol** **Edon Berisha**
Studienjahr	**3. Studienjahr**
Titel der Studienarbeit	**SIAN** **Software zur Identifikation von Applikationsklassen in NetFlow**
Examinator	***Eduard Glatz***
Themengebiet	***Netzwerkdaten, Software***
Projektpartner	***ETHZ, Bernhard Tellenbach***
Institut	**Diverses**

Kurzfassung der Studienarbeit

Das Ziel dieser Semesterarbeit besteht darin, das vorhandene Framework um ein Modul zu erweitern, das die Identifikation verschiedener Applikationsklassen unterstützt. Die verschiedenen Heuristiken zur Identifikation der Applikationsklassen wurden von der Literatur entnommen. Zur Umsetzung dieser Heuristiken ist die Wahl von effizienten Datenstrukturen und Algorithmen eminent wichtig.

Im Prototyp dieser Arbeit werden Intervalle von 10 Minuten gebildet. Ein 10-Minuten-Intervall zählt etwa 20 Millionen Flows. Flows enthalten Informationen über den IP-Datenstrom. Von diesem Datenstrom werden in den Flows nur die Header-Informationen wie Source- und Destination-IP gespeichert. Um diese Daten verarbeiten zu können, stehen ein leistungsfähiges Software-Framework sowie ein moderner Linux-Cluster zur Verfügung.

Für die Analyse der Heuristiken wurden verschiedene Literaturangaben benutzt. Diese wurden dann für diesen Prototyp erweitert oder angepasst. Aufgrund der riesigen zu verarbeitenden Datenmenge, die ein 10-Minuten-Intervall mit sich bringt, ist auch eine Analyse der Zeit- und Raumkomplexität bezüglich Datenstrukturen und Algorithmen sehr wichtig, damit sich die Speicher- und Zeitzahlen in einem akzeptablen Rahmen bewegen.

Der Benutzer kann das SIAN-Modul im Framework alleine oder mit anderen Modulen kombiniert starten. Per Kommandozeilen-Befehl kann man beim Aufruf der Applikation als Programmparameter die Start- und die Stop-Zeit der zu verarbeitenden Daten bzw. Flows mitgeben.

Der SIAN-Prototyp wertet dann in den erwähnten 10-Minuten-Intervallen die Flows gemäss den verschiedenen Applikationsklassen, welche in den dokumentarisch vorhandenen Heuristiken beschrieben sind, aus und schreibt das Statistik-Resultat in eine CSV-Datei.

Erreicht wurde ein kompilierbarer Prototyp, der zwar gestartet werden kann, aber noch keine Tests betreffend Raum und Zeitkomplexität enthält.

4. Management Summary

4.1. Ausgangslage

4.1.1. Was soll mit dem Projekt erreicht werden?

An der ETH Zürich werden zu Forschungszwecken Datenanalysen an realen Verkehrsdaten von der Firma SWITCH analysiert. Dazu steht eine grosse Sammlung (ca. 29TB komprimiert) von CISCO-NetFlow-Daten zur Verfügung, die fortlaufend mit den neusten Verkehrsinformationen ergänzt wird. Um diese Daten verarbeiten zu können, stehen ein leistungsfähiges Software-Framework sowie ein moderner Linux-Cluster zur Verfügung.

In diesem Framework fehlt aber die Identifikation der einzelnen Applikationsklassen. Dieses Projekt dient der Erarbeitung des entsprechenden Moduls, welches diese Applikationsidentifikation (P2P, Web, DNS, Mail, Messenger, Attacken, etc.) vornimmt. Im Rahmen der vorliegenden Semesterarbeit soll ein solider Prototyp erstellt werden.

4.1.2. Was beinhalten andere, ähnliche Projekte?

Aus den Literaturnachschlagewerken geht hervor, dass sich schon andere Universitäten mit der Problematik eines Projekts, wie es oben beschrieben ist, auseinandergesetzt haben und ihre eigenen Evaluationen erstellt haben. Das SIAN-Projekt nimmt diese Erkenntnisse als Grundlage und entwickelt ein eigenes Modul, das speziell an das bestehende Framework und dessen Bedürfnisse angepasst wurde. Auch wurden einige Heuristiken für das Framework entsprechend erweitert oder modifiziert.

4.1.3. Was macht das Projekt besonders interessant?

Dieses Projekt hat für uns zwei spezielle Anreize:

Es ist für eine konkrete Anwendung bestimmt, und es wird anschliessend auch genutzt.

Und es unterscheidet sich in der Thematik von einer "Routinearbeit" im Sinne von "Desktopapplikation mit Benutzeroberfläche und einfacher Datenbankanbindung". Eine besondere Herausforderung besteht darin, selbst eine Speicher- und Verarbeitungsideologie zu entwickeln, um sehr grossen Daten in einer akzeptablen Zeit verarbeiten zu können. Das bedeutet, dass die Raum- und Zeitkomplexität in diesem Projekt eine sehr grosse Rolle spielt. Die dafür geeigneten Datenstrukturen und Algorithmen zu analysieren und zu implementieren machten dieses Projekt äusserst interessant.

4.1.4. Welche Ziele wurden gesteckt? (Kann-Ziele, Muss-Ziele)

In dieser Arbeit ging es vor allem darum, aus der Aufgabenstellung eine grobe Analyse zu erstellen. Nachfolgend musste eine detaillierte Untersuchung der verschiedenen Heuristiken abgeleitet und eine eigene Idee von Speicherstrukturen entwickelt werden, welche eine akzeptable Raum- und Zeitkomplexität nachweist und den Projektanforderungen gerecht wird. Die Arbeit sollte in einen Prototypen münden, der Flow-Daten einliest, diese nach den Heuristiken auswertet und das daraus resultierende Auswertungsergebnis in eine CSV-Datei schreibt. Die Auswertung muss in 10-Minuten-Intervallen erfolgen. Der User kann per Kommandozeile die Start- und Stopzeit der auszuwertenden Flow-Daten eingeben.

Erweiterungsmöglichkeiten wurden so vorbereitet, dass die Intervalle ohne grösseren Codeaufwand, auf verschiedene Zeitabstände geändert werden können. Eine andere Erweiterungsmöglichkeit besteht in der Unterscheidung zwischen IPv4 und IPv6, die zu einem späteren Zeitpunkt vorgenommen werden könnte. Im vorliegenden Prototypen werden IPv4-Pakete ausgewertet und IPv6-Pakete werden verworfen. Auch eine allfällige Erweiterung der Heuristiken kann ohne grösseren Codeaufwand realisiert werden.

4.2. Vorgehen

4.2.1. Abwicklung

Das Projekt haben wir nach dem RUP-Modell realisiert. So haben wir in der Inception-Phase eine Anforderungsanalyse und einen Machbarkeitstest durchgeführt. Dieser beinhaltete ein kleines Programm, welches in der Lage war, zusätzliche NetFlow-Informationen zu verarbeiten. In der nachfolgenden Elaboration-Phase haben wir erste Lösungsentwürfe entwickelt, welche das Speicher- und Laufzeitverhalten der Datenstrukturen und Algorithmen beschreiben. Daraufhin implementierten wir einen umfassenden Pseudo-Code, welcher als Grundlage für die Entwicklung in der Construction-Phase diente. In der Transition-Phase machten wir uns Gedanken über Modul-Tests. Wir sind zum Schluss gekommen, dass der sinnvollste Test für unsere Applikation so aufgebaut werden muss, dass jede Regel (H1-F10) mit Hilfe von vorpräparierten Daten, welche den Ausschlag einer Regel provozieren, geprüft wird. Dank den Treffen mit dem Projektleiter und dem Industriepartner, die wöchentlich stattfanden, konnten wir unsere erarbeiteten Projektzwischenergebnisse regelmässig vorlegen, Varianten diskutieren, die weiteren Schritte besprechen und stets eine effiziente und qualitativ hohe Lösung finden.

4.2.2. Teilschritte

Inception

Schritt 1: Einarbeitung in das bestehende Framework und dessen Arbeitsumgebung.

Schritt 2: Analyse der Aufbauweise des Flow-Formats und entsprechenden Daten.

Schritt 3: Überprüfung der in der Literatur angegebenen Heuristiken.

Elaboration

Schritt 4: Behandlung folgender Fragen mit dem Industriepartner und dem Projektverantwortlichen:

welche Heuristiken sollten auf welche Weise für das Modul übernommen werden und welche Erweiterungen oder Anpassungen müssten gemacht werden.

Schritt 5: Erster Entwurf in Form von Pseudo-Code.

Schritt 6: Erstellung einer Zeit- und Raumkomplexitätsanalyse der Algorithmen und Datenstrukturen.

Construction

Schritt 7: Implementierung des Moduls SIAN.

Transition

Schritt 8: Testen des Moduls mit Testdaten.

Schritt 9: Einbinden des Moduls in das Framework und testen mit realen Daten.

Schritt 10: Abgabe an den Projektverantwortlichen.

4.2.3. Wer war involviert?

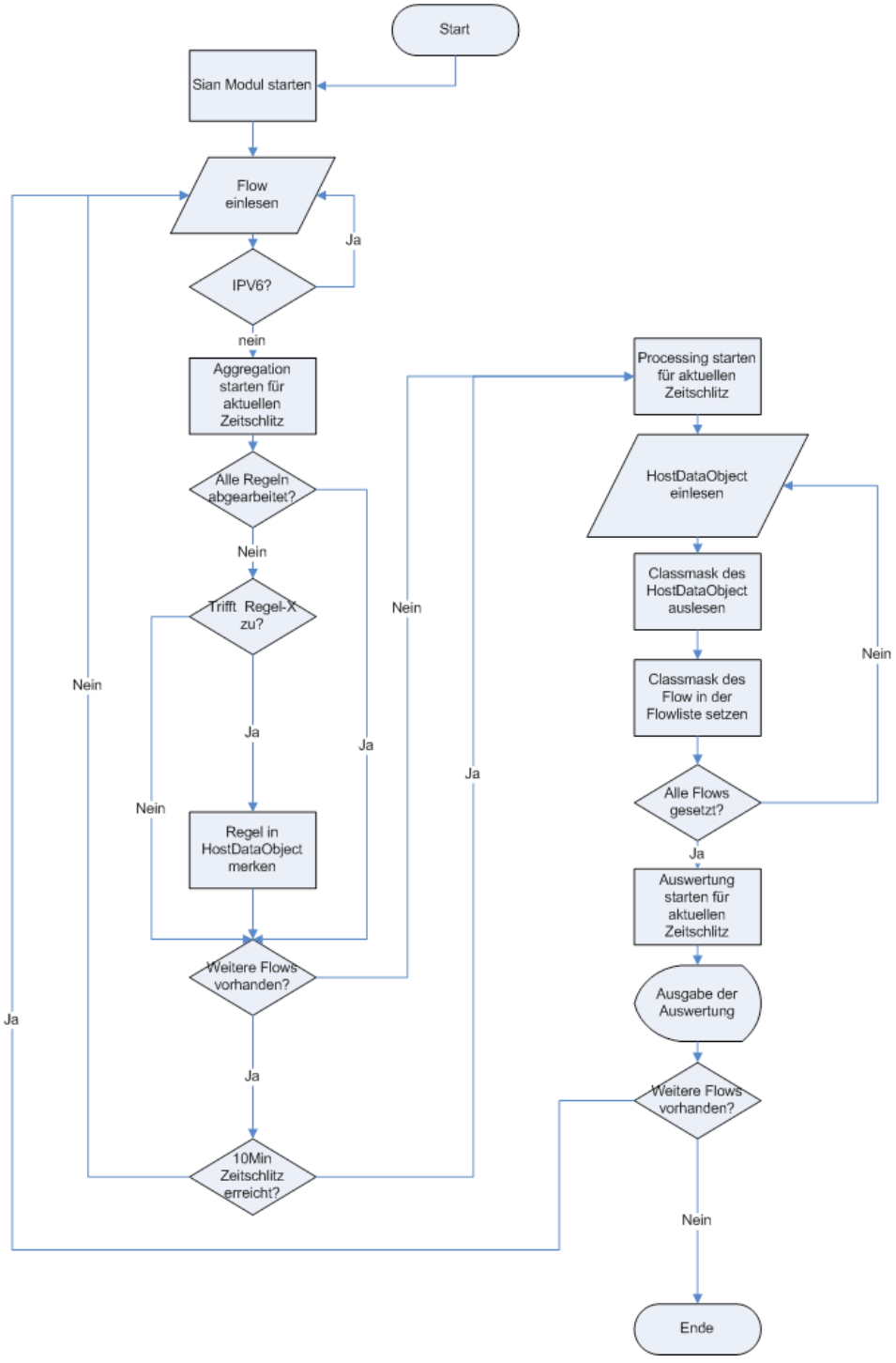
Funktion	Name
Betreuer	Eduard Glatz
Industriepartner	ETHZ, Bernhard Tellenbach
Entwickler	Selim Akyol
Entwickler	Edon Berisha

4.3. Ergebnisse

4.3.1. Resultat des Projekts

Das Resultat dieser Semesterarbeit ist ein stabiler Prototyp, das NetFlow-Daten einlesen kann und diese nach den Heuristiken analysiert und auswertet. Die gesamte Auswertung wird in 10-Minuten Intervallen unterteilt. Jede Intervall-Auswertung wird in die CSV-Datei geschrieben. Diese CSV-Datei bildet dann den Abschluss dieses Moduls mit den notwendigen Ergebnisinformationen. Das SIAN-Modul kann ohne Probleme mit den anderen Modulen des Frameworks kombiniert werden. Auch kann der Benutzer mittels Kommandozeile die entsprechende Start- und Stopzeit der Auswertung an das Modul mitgeben.

Das Ablaufprinzip des Moduls wird anhand des folgenden Flussdiagramms veranschaulicht:



4.3.2. Was musste erarbeitet werden, was war schon vorhanden?

Die verschiedenen Heuristiken konnten wir der Literatur entnehmen. Auch das Framework war schon vorhanden. Für diese Arbeit haben wir die Heuristiken an das Framework angepasst und wenn nötig erweitert und die entsprechenden Analysen wie Zeitkomplexität und Raumkomplexität geschrieben.

Diese Analysen bildeten die Grundlage für die Erarbeitung des entsprechenden Datenstrukturen und Algorithmen. Schlussendlich haben wir die Applikation implementiert.

4.3.3. Schätzung und Zielerreichung

Ziel dieses Projekts war es, das Framework um ein Modul zu erweitern, das die verschiedenen Applikationsklassen mittels Heuristiken identifiziert. Dieses Ziel wurde auf theoretischer Ebene erfüllt. Auf der praktischen Ebene fehlten bis zum Schluss des Projekts die Tests betreffend der Raum- und Zeitkomplexität.

Anfänglich wurde das Projekt mit zwei Wochen Reserve als machbar beachtet. Am Schluss des Projekts fehlten nur noch die Tests betreffend der oben erwähnten Zeit- und Raumkomplexität.

4.3.4. Abweichungen

Ausser den fehlenden Tests betreffend der Zeit- und Raumkomplexität wurden alle Ziele erreicht.

4.3.5. Kosten

Der Zeitaufwand des Projektes wurde zu Beginn auf ca. 480 Stunden geschätzt. Am Ende dieser Semesterarbeit sind wir auf ein Total von 620 Stunden gekommen. Für die komplette Fertigstellung des Projektes wird mit nochmals ca. 200 Stunden gerechnet.

4.3.6. Lernpunkte aus der Durchführung des Projekts

Bei dieser Semesterarbeit konnten wir unsere C++ Kenntnisse sehr vertiefen. Neues Wissen konnten wir uns nicht nur im Netzwerkbereich, sondern auch auf dem Gebiet der Analysen über Raum- und Zeitkomplexität über die verschiedenen Algorithmen und Datenstrukturen aneignen.

4.4. Ausblick

4.4.1. Offene Punkte

Der einzige offene Punkt sind die fehlenden Tests betreffend der Zeit und Raumkomplexität und die Beseitigung der letzten Runtime Errors.

4.4.2. Was ist weiter zu tun

Nach Abschluss dieser Semesterarbeit sind sicher die fehlende Tests nachzuführen.

Auch die IPv6-Pakete müssen zustäzlich zu den IPv4 Paketen bearbeitet werden können.

Denkbar wäre auch, dass die Intervallgrösse ebenfalls per Kommandozeile beim Aufruf des Programms eingegeben werden kann.

5. Technischer Bericht

5.1. Einleitung und Übersicht

In dieser Semesterarbeit geht es darum, ein bestehendes Framework, das den Netzwerkverkehr analysiert, um ein Modul zu erweitern.

Das aktuelle Framework besitzt mehrere Module, die CISCO-NetFlow-Pakete von Routern des Providers SWITCH in einer Datenbank speichern und für Forschungsarbeiten untersuchen.

Ein NetFlow-Paket beinhaltet nur Header-Informationen einer Verbindung. Diese Informationen wurden in den letzten 6 Jahren kontinuierlich gespeichert. Das angesammelte Datenvolumen beläuft sich auf ca. 29 TB.

In diesem Framework fehlt zur Zeit noch ein Modul, welches die einzelnen Flow-Records nach verschiedenen Heuristiken analysiert und auswertet.

Das Ziel dieser Semesterarbeit ist die Entwicklung dieses Moduls. Das neue Modul namens SIAN (Software zur Identifikation von Applikationsklassen in NetFlow) soll die einzelnen NetFlow-Records analysieren und gemäss den dokumentierten Heuristiken auswerten.

Vielfällige Literatur zu den Heuristiken wurde uns zur Verfügung gestellt, was die Aufteilung und Analyse der verschiedenen Applikationsklassen erleichterte.

Anhand der Literatur konnten wir unsere Heuristikauswertung an die Vorgaben und Nutzungszwecke des Frameworks anpassen.

5.2. Ergebnisse

Diese Semesterarbeit gliedert sich in zwei grosse Teile:

Der 1. Teil besteht aus der Analyse der verschiedenen Heuristiken und des aktuellen Frameworks.

Der 2. Teil hat das Implementieren der bearbeiteten Heuristiken und das Testen zum Thema.

1 Teil: Analyse der Heuristiken

Die Analyse der Heuristiken sowie die Einarbeitung in das bestehende Framework haben uns viel Zeit gekostet. Die Erkenntnisse aus der Literatur erlaubten uns, 15 verschiedene Applikationsklassen herauszukristallisieren. Wir unterscheiden zuerst zwischen zwei Grossgruppen, nämlich zwischen P2P- und NonP2P-Verkehr. Der NonP2P-Verkehr wird nochmals in 10 weitere Applikationsklassenn nämlich Web (2), DNS, Mail, Messenger, Gaming, FTP, NonP2P-Ports, Attacks und unclassified Flows, unterteilt. Diese werden auch "false positive flows" genannt. Es handelt sich also um Flows, welche zwar zuvor als P2P identifiziert wurden, die jedoch aufgrund einer zutreffenden F-Regel keine gültige P2P-Eigenschaft mehr aufweisen. Einige Heuristiken, die das Verhalten über mehrere Tage beobachten sollten, wurden so angepasst, dass sie in den 10-Minuten-Intervallen realitätsnahe Auswertungen ausführen können. Mit dieser Anpassung wurde namentlich die Dauer einer Heuristik von mehreren Tagen oder Stunden auf 10 Minuten gekürzt. Wichtig ist, dass eine Verbindung bidirektional ist, das heisst also, dass beim Verbindungsaufbau von der Source zur Destination eine bidirektionale Kommunikation zu Stande kommen sollte. Im theoretischen Fall würde dies immer stimmen. Dass das in der Realität aber nicht immer so ist, beweist nur schon die Tatsache, dass es in grossen IT-Netzwerken immer wieder zu Unterbrüchen kommt. Aus diesem Grund speichern wir nicht nur Host-Informationen über die Source, sondern auch über die Destination in der Annahme, dass die Möglichkeit besteht, dass kein Antwort-Flow zurückkommt. In der Regel aber haben wir mindestens 2 Flows pro Verbindung.

Um das Verständnis zu vereinfachen, sind auf den folgenden Seiten alle Regeln in Pseudocodeform aufgeschrieben.

H1 TCP/UPD IP Paare

Aus [1] : „ ***H1: TCP/UDP IP Pairs***:(K),(P). This rule exploits the fact that many P2P applications use TCP for data transfer and UDP for signaling traffic. Source and destination IP pairs, which concurrently use TCP and UDP are therefore marked as P2P hosts. All flows to and from these hosts are marked as potential P2P flows. Concurrent here means usage of TCP and UDP within the 10 minutes interval. Karagiannis identified some non-P2P applications which show a similar behavior, such as netbios, dns, ntp and irc (Table 3 in [4]). UDP flows from these applications are excluded from this heuristic based on their port-numbers. “

Unsere Interpretation:

Viele Peer-To-Peer Applikationen nützen das TCP-Protokoll um Daten auszutauschen und das UDP-Protokoll um einen Datenverkehr zu signalisieren. Quell- und Ziel-IP – Paare, die nebenläufig TCP und UDP benutzen werden deshalb als P2P-Verkehr markiert. Mit nebenläufig ist hier gemeint, dass in einem 10 Minuten – Intervall TCP- und UDP-Pakete vorkommen.

```
//Aggregatorphase
if(!wkNonP2PportsArray[((*flow)->port_src)])
{
        srcHostDataObject->setH1F5F6Iterator
        (srcHostDataObject->getH1F5F6().find(*((uint32_t*)((*flow)->ip_dst))));

        if(srcHostDataObject->getH1F5F6Iterator() == srcHostDataObject->getH1F5F6().end()) {
                srcHostDataObject->getH1F5F6().insert
                (IPHashMap_pair(*((uint32_t*)((*flow)->ip_dst)), 0));
        }

        // Je nach Protokollart, Bitmaske entsprechend setzen.
        // Bitmaske              für TCP:         1. Bit (Wert 1)
        //                       für UDP:         2. Bit (Wert 2)
        //                       für Beide:       3. Bit (Wert 4)
        // Protokoll-Nummern: TCP = 6, UDP = 17
        if(srcHostDataObject->getBitMaskOfH1F5F6OnIP(*((uint32_t*)((*flow)->ip_dst)) != 3)) {
                tempBitMask = srcHostDataObject->
                getBitMaskOfH1F5F6OnIP(*(uint32_t*)(*flow)->ip_dst);
                if((*flow)->protocol == 6)
                {
                        srcHostDataObject->
                        setBitMaskInH1F5F6ForIP(*((uint32_t*)((*flow)->ip_dst)), tempBitMask || 1);
                } else if((*flow)->protocol == 17) {
                        srcHostDataObject->
                        setBitMaskInH1F5F6ForIP(*((uint32_t*)((*flow)->ip_dst)), tempBitMask || 2);
                        }
                }
        }
//Processing Phase
//H1
if(srcHostDataObject != NULL) {
        tempBitMask = srcHostDataObject->getBitMaskOfH1F5F6OnIP((*flowListIterator)->getDstIP());
        if(tempBitMask & bitMaskArray[3] > 0){
                tempBitMask = (*flowListIterator)->getClassmask();
                tempBitMask = tempBitMask | bitMaskArray[1];
                (*flowListIterator)->setClassmask(tempBitMask);
        }
}
```

H2 P2P Ports

Aus [1] : „ ***H2: P2P Ports***:(P). Even though many P2P applications choose arbitrary ports for their communication, approx. one third of all P2P traffic can still be identified by known P2P destination port numbers [2]. Furthermore, it seems disadvantageous for non-P2P applications to deliberately use well known P2P ports for their services, since traffic on these ports is often blocked by traffic filters in some networks. Flows to and from port numbers listed in Table 3 of [2], enriched with additional P2P ports, are marked as potential P2P traffic. “

Unsere Interpretation:

Obwohl viele P2P-Applikationen heutzutage beliebige Ports verwenden, können dennoch ca. 1/3 vom gesamten P2P-Datenverkehr aufgrund der typischen P2P-Zielports identifiziert werden. Verkehrsflüsse, die diese Zielports verwenden, werden als P2P-Fluss markiert.

```
//Aggregatorphase
-
//Processing Phase
//H2
                if(wkP2PportsArray[(*flowListIterator)->getDstPort()]) {
                        tempBitMask = (*flowListIterator)->getClassmask();
                        tempBitMask = tempBitMask | bitMaskArray[2];
                        (*flowListIterator)->setClassmask(tempBitMask);
                }
```

H3 Port Benutzung

Aus [1] : „ ***H3: Port Usage***:(P). In normal application, the operating system assigns ephemeral port numbers to source ports when initiating connections. These numbers are often iterating through a configured ephemeral port space. It is very unusual, that the same port numbers are used within short time periods. This however can be the case for P2P applications with fixed ports assigned for signaling traffic or data transfer. If a source port on a host is repeatedly used within 60 seconds, the host is marked as P2P host, and all flows to and from this host are marked as potential P2P flows. “

Unsere Interpretation:

Wenn der gleiche Quellport wiederholend auf einem Host innerhalb von 60 Sekunden gebraucht wird, so ist es ein P2P-Host und alle Flüsse von und zu diesem Host sind P2P-Flüsse.

```
//Aggregatorphase
srcHostDataObject->setH3F4Iterator(srcHostDataObject->getH3F4().find((*flow)->port_src));

if(srcHostDataObject->getH3F4Iterator() == srcHostDataObject->getH3F4().end()) {
        srcHostDataObject->getH3F4().insert(PortHashMap_pair((*flow)->port_src, 0));
}
if(srcHostDataObject->getBitMaskOfH3F4OnPort((*flow)->port_src) & bitMaskArray[1] == 0) {

        srcHostDataObject->setSrcPortUsedAtTimeHashMapIterator(
        srcHostDataObject->getSrcPortUsedAtTimeHashMap().find((*flow)->port_src));

        if(srcHostDataObject->getSrcPortUsedAtTimeHashMapIterator() !=
                srcHostDataObject->getSrcPortUsedAtTimeHashMap().end()) {
                if(srcHostDataObject->getSrcPortUsedAtTimeHashMapIterator()->second.second) {
                        if(abs((int)((*flow)->start_ms -
                        srcHostDataObject->
                        getSrcPortUsedAtTimeHashMapIterator()->second.first)) <= 60000) {
                                tempBitMask =
                                srcHostDataObject->getBitMaskOfH3F4OnPort((*flow)->port_src);
                                tempBitMask = tempBitMask | 1;
                                srcHostDataObject->
                                setBitMaskInH3F4ForPort((*flow)->port_src, tempBitMask);
                        }
                }
        }
}
//Processing Phase
if(srcHostDataObject != NULL) {
        srcHostDataObject->setH3F4Iterator(
        srcHostDataObject->getH3F4().find((*flowListIterator)->getSrcPort()));
        if(srcHostDataObject->getH3F4Iterator() != srcHostDataObject->getH3F4().end()) {
                tempBitMask = srcHostDataObject->getH3F4Iterator()->second;
                if(tempBitMask & bitMaskArray[1] > 0) {
                        tempBitMask = (*flowListIterator)->getClassmask();
                        tempBitMask = tempBitMask | bitMaskArray[3];
                        (*flowListIterator)->setClassmask(tempBitMask);
                }
        }
}
```

H4 P2P IP/Port Paare

Aus [1] : „ ***H4: P2P IP/Port Pairs***:(K). If listening ports on peers in P2P networks are not well known in advance, they are typically propagated to other peers by some kind of signaling traffic (e.g. an overlay network). This means that each host connecting to such a peer will connect to this agreed port number, using a random, ephemeral source port. As noted by Karagiannis, P2P peers usually maintain only one connection to other peers, which means that each endpoint (IP,port) has at least the same number of distinct IP addresses (#sIP) and number of distinct ports (#sPort) connected to it. If #sPort- #sIP<2 and #sIP>5, the host is considered as P2P host, and all flows to and from this host are marked as potential P2P. “

Unsere Interpretation:

P2P-Peers halten normalerweise nur eine einzelne offene Verbindung zu einem anderen Peer offen, was heisst, dass jeder Endpunkt (IP, Port) eine gleich grosse IP-Liste mit eindeutigen Quell-IP-Adressen und eindeutigen Quell-Ports beinhaltet. Wenn also die Anzahl eindeutiger Quell-Port-Nummern – Quell-IP’s < 2 ist und die Anzahl eindeutigerQuell-IP’s > 5, dann wird dieser Host als P2P-Host vermerkt und alle Flüsse von und zu diesem als P2P-Fluss markiert.

```
//Aggregatorphase
endPointKey endPointObject((uint32_t*)((*flow)->ip_src), (uint16_t*)((*flow)->port_src));
ipPort_pair ipPort = make_pair(*(uint32_t*)(*flow)->ip_src, (*flow)->port_src);

interval->setEndPointKeyHashMapIterator(interval->getEndPointKeyHashMap()->find(ipPort));

if(interval->getEndPointKeyHashMapIterator() == interval->getEndPointKeyHashMap()->end()) {
        EndPointKeyHashMap_pair endPointPair = make_pair(ipPort, endPointObject);
        interval->getEndPointKeyHashMap()->insert(endPointPair);
        interval->setEndPointKeyHashMapIterator(interval->getEndPointKeyHashMap()->find(ipPort));
}

((interval->getEndPointKeyHashMapIterator())->second).setConnectedIPHashMapIterator(

(((interval->getEndPointKeyHashMapIterator())->second).getConnectedIPHashMap()).find(
*((uint32_t*)((*flow)->ip_dst))
)
);

if((interval->getEndPointKeyHashMapIterator())->second.getConnectedIPHashMapIterator() ==
(interval->getEndPointKeyHashMapIterator())->second.getConnectedIPHashMap().end()) {
        (interval->getEndPointKeyHashMapIterator())->second.insertIP(*(uint32_t*)(*flow)->ip_dst);
}

((interval->getEndPointKeyHashMapIterator())->second).setConnectedPortHashMapIterator(

(((interval->getEndPointKeyHashMapIterator())->second).getConnectedPortHashMap()).find(
((*flow)->port_dst)));

if((interval->getEndPointKeyHashMapIterator())->second.getConnectedPortHashMapIterator() ==
(interval->getEndPointKeyHashMapIterator())->second.getConnectedPortHashMap().end()) {
        (interval->getEndPointKeyHashMapIterator())->second.insertPort((*flow)->port_dst);
}


//Processing Phase
ipPort_pair ipPort = make_pair((*flowListIterator)->getSrcIP(), (*flowListIterator)->getSrcPort());
interval->setEndPointKeyHashMapIterator(interval->getEndPointKeyHashMap()->find(ipPort));
if((interval->getEndPointKeyHashMapIterator()->second.countOfConnectedPorts()) - (interval->
getEndPointKeyHashMapIterator()->second.countOfConnectedIPs()) < 2
&& (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedIPs()) > 5){
        interval->getEndPointKeyHashMapIterator()->second.setH4(true);
        tempBitMask = (*flowListIterator)->getClassmask();
        tempBitMask = tempBitMask | bitMaskArray[4];
        (*flowListIterator)->setClassmask(tempBitMask);
}
```

H5 Undefined P2P

Aus [1] : „ ***H5: unclassified, long flow***:(P) After removing well known applications from the unclassified flows, we mark remaining unclassified flows which carry more than 1 MB of data in one direction or have connection durations of over 10 minutes as P2P flows. This rule is based on Perenyis heuristic 6, even though we believe it is a very weak rule. However, there is a large probability, that such long flows in fact are P2P flows. “

Unsere Interpretation:

Datenflüsse die nicht als H1, H2, H3 oder H4 markiert wurde und eine Grösse von über 1MB haben und eine Dauer von mehr als 10 Min haben werden als H5 markiert.

```
//Aggregatorphase
if((srcHostDataObject->getBitMaskOfH1F5F6OnIP(*(uint32_t*)(*flow)->ip_src) != 4) &&
(srcHostDataObject->getBitMaskOfH3F4OnPort((*flow)->port_src) != 2) &&
(srcHostDataObject->getBitMaskOfH4F3F9OnEndPoint(endPointObject) != 2)) {
        if(((*flow)->duration_ms >= 600000) || ((*flow)->bytes > 1048576)) {
                srcHostDataObject->setH5(true);
        }
}
//Processing Phase
if(srcHostDataObject != NULL) {
        if(srcHostDataObject->getH5()) {
                tempBitMask = (*flowListIterator)->getClassmask();
                tempBitMask = tempBitMask | bitMaskArray[5];
                (*flowListIterator)->setClassmask(tempBitMask);
        }
}
```

F1 Web IP/Port Paare

Aus [1] : „ ***F1: Web IP/Port Pairs***:(K). Web traffic on the other hand typically uses multiple connections to one server. For this reason hosts are marked as web-hosts, if the difference between #sPort and #sIP connected to an endpoint (IP,port) is larger than 10, the ratio between #sPort and #sIP is larger than two and at least 10 different IPs are connected to this endpoint (#sPort-#sIP>10 and #sPort/#sIP>2 and #sIP>10). All flows with http port numbers (80, 443, 8080) to and from these webhosts are then marked as web traffic. “

Unsere Interpretation:

Im Gegensatz zu P2P führt ein Webserver mehrere Verbindungen. Wenn die Differenz zwischen der Anzahl eindeutiger Portnummern und eindeutiger IP-Nummern, die zu einem Endpunkt verbunden sind, grösser als 10 ist, der Quotient von Eindeutiger Portnummern/ Eindeutige IP-Nummern grösser als 2 und mindestens 10 verschiedene IP’s zu diesem Endpunkt verbunden sind (**#sPort-#sIP>10 && #sPort/#sIP && #sIP>10**), dann wird dieser Endpunkt als Webhost markiert. Alle Flüsse mit den Portnummern 80, 443 und 8080 zu und von diesem Webhost sind als Webflüsse markiert.

```
//Aggregatorphase
keine, da diese gleich ist wie H4
//Processing Phase
if(srcHostDataObject != NULL) {
        if((abs((int)(interval->getEndPointKeyHashMapIterator()->second.countOfConnectedPorts()-
        (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedIPs()) > 10)
        && (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedPorts())/(
        interval->getEndPointKeyHashMapIterator()->second.countOfConnectedIPs()) > 2
        && (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedIPs() > 10)))) {
                tempBitMask = (*flowListIterator)->getClassmask();
                tempBitMask = tempBitMask | bitMaskArray[6];
                (*flowListIterator)->setClassmask(tempBitMask);
        }
}
```

F2 Web

Aus [1] : „ ***F2: Web***:(P). To further identify web traffic, we followPerenyi's heuristic number 2, taking advantage of the fact that web clients typically not only use multiple, but even parallel connections to webservers. Hosts with parallel connections to a http port are considered as webservers. All flows to and from web servers on http ports are marked as web traffic. "

Unsere Interpretation:

Webclients benutzen normalerweise nicht nur mehrere, sondern sogar parallele Verbindungen zu einem Webserver. Hosts (Clients) mit parallelen Verbindungen zu einem Http-Port werden als Webhost markiert und alle Flüsse von und zu einem Webserver als Webflüsse.

```
//Aggregatorphase
if(webPortsArray[(*flow)->port_src]) {
        srcHostDataObject->setConnectionToIPWithHttpPortHashMapIterator(
        srcHostDataObject->getConnectionToIPWithHttpPortHashMap().find(*(uint32_t*)(*flow)->ip_dst));
        if(srcHostDataObject->getConnectionToIPWithHttpPortHashMapIterator() !=
        srcHostDataObject->getConnectionToIPWithHttpPortHashMap().end()) {
        if(srcHostDataObject->wasAlreadyConnectedToIPWithHttpPort(*(uint32_t*)(*flow)->ip_dst)) {
                srcHostDataObject->setF2(true);
        } else {
                srcHostDataObject->setHttpFlagForIPInMap(true, *(uint32_t*)(*flow)->ip_dst);
        }
        } else {
                srcHostDataObject->getConnectionToIPWithHttpPortHashMap().insert(
                connectionToIPWithHttpPortHashMap_pair(*(uint32_t*)(*flow)->ip_dst, true));
        }
} else if(webPortsArray[(*flow)->port_dst]){
                dstHostDataObject->setConnectionToIPWithHttpPortHashMapIterator(
                dstHostDataObject->getConnectionToIPWithHttpPortHashMap().find(
                *(uint32_t*)(*flow)->ip_src));

        if(dstHostDataObject->getConnectionToIPWithHttpPortHashMapIterator() !=
        dstHostDataObject->getConnectionToIPWithHttpPortHashMap().end()) {
                if(dstHostDataObject->wasAlreadyConnectedToIPWithHttpPort(*(uint32_t*)(*flow)->ip_src)) {
                        dstHostDataObject->setF2(true);
                } else {
                        dstHostDataObject->setHttpFlagForIPInMap(true, *(uint32_t*)(*flow)->ip_src);
                }
        } else {
                dstHostDataObject->getConnectionToIPWithHttpPortHashMap().insert(
                connectionToIPWithHttpPortHashMap_pair(*(uint32_t*)(*flow)->ip_src, true));
        }
}//Processing Phase
if(srcHostDataObject != NULL) {
        if(srcHostDataObject->getF2()) {
                tempBitMask = (*flowListIterator)->getClassmask();
                tempBitMask = tempBitMask | bitMaskArray[7];
                (*flowListIterator)->setClassmask(tempBitMask);
        }
}
if(((*flowListIterator)->getClassmask() & bitMaskArray[7]) == 0) {
        if(dstHostDataObject != NULL) {
                if(dstHostDataObject->getF2()) {
                        tempBitMask = (*flowListIterator)->getClassmask();
                        tempBitMask = tempBitMask | bitMaskArray[7];
                        (*flowListIterator)->setClassmask(tempBitMask);
                }
        }
}
```

F3 DNS

Aus [1] : „ ***F3: DNS***:(K). Traditional services like dns sometimes use equal source port and destination port numbers. As suggested by Kargiannis, we mark endpoints (IP,port) as non-P2P, if it includes flows with equal source- and destination port and port numbers smaller than 501. All flows to and from this endpoint are then
marked as non-P2P traffic. “

Unsere Interpretation:

DNS benutzt manchmal die gleiche Quell- und Zielportnummer. Wir markieren Endpunkte (IP, Port) als non-P2P (DNS), wenn sie Flüsse mit gleichen Quell- und Zielportnummern, die kleiner als 501 sind, beinhalten.

```
//Aggregatorphase
if((*flow)->port_src == (*flow)->port_dst && ((*flow)->port_src < 501 || (*flow)->port_dst < 501)) {
        interval->getEndPointKeyHashMap()->find(ipPort)->second.setF3(true);
}


//Processing Phase
if(srcHostDataObject != NULL) {
        tempBitMask = srcHostDataObject->getBitMaskOfH4F3F9OnEndPoint(endPointObject);
        if(interval->getEndPointKeyHashMap()->find(ipPort)->second.isF3()) {
                tempBitMask = (*flowListIterator)->getClassmask();
                tempBitMask = tempBitMask | bitMaskArray[8];
                (*flowListIterator)->setClassmask(tempBitMask);
        }
}
```

F4 Mail

Aus [1] : „ ***F4: Mail***:(K). Hosts receiving traffic on mail ports (smtp, pop, imap) and in the same analysis interval also initiate connections to port 25 on other hosts are considered to be mailservers. All flows to and from mailservers are marked as mail traffic. “

Unsere Interpretation:

Hosts, die Datenverkehr am Mailport empfangen und neue Verbindungen zu anderen Hosts mit der Portnummer 25 starten markieren wir als Mailserver. Alle Flüsse von und zu diesem Mailserver sind Mailflüsse.

```
//Aggregatorphase
if(mailPortsArray[(*flow)->port_src]) {
        if((*flow)->port_src == 110) {
                srcHostDataObject->setValueInMailArrayAtPosition(true, 0);
        } else if((*flow)->port_src == 143 || (*flow)->port_src == 993 || (*flow)->port_src == 220) {
                srcHostDataObject->setValueInMailArrayAtPosition(true, 1);
        } else if((*flow)->port_src == 587 || (*flow)->port_src == 465) {
                srcHostDataObject->setValueInMailArrayAtPosition(true, 2);
        } else if((*flow)->port_dst == 25) {
                srcHostDataObject->setValueInMailArrayAtPosition(true, 3);
        }
}
//Processing Phase
if(srcHostDataObject != NULL) {
        if(srcHostDataObject->getMailArrayAtPosition(3)) {
                if(srcHostDataObject->getMailArrayAtPosition(0) ||
                srcHostDataObject->getMailArrayAtPosition(1) ||
                srcHostDataObject->getMailArrayAtPosition(2)) {
                        tempBitMask = tempBitMask | bitMaskArray[9];
                        (*flowListIterator)->setClassmask(tempBitMask);
                }
        }
}
```

F5 Messenger

Aus [1] : „ ***F5: Messenger***:(K). Popular messenger and chat servers (icq, yahoo, msn, jabber, irc) tend to have long uptimes and rarely change IP addresses, especially when maintained by commercial providers such as Microsoft and Yahoo. To improve the accuracy of the results, in this heuristic we therefore take advantage of the whole 20 day long dataset. Hosts, connected to by at least 10 different IPs on well known messenger ports within a period of at least 10 days, are marked as messenger servers. All traffic to and from these hosts on known messenger ports is classified as messenger traffic. “

Unsere Interpretation:

Hosts, die mit mindestens 10 verschiedenen IP’s und bekannten Messenger-Ports in einer Mindestzeitperiode von 10 Tagen verbunden sind, markieren wir als Messenger-Server. Flüsse von und zu diesem Server sind Messenger-Flüsse. Wir vermerken hier, dass wir die Überprüfung von 10 Tagen nicht machen können.

```
//Aggregatorphase
keine, da diese gleich ist wie H1
//Processing Phase
if(srcHostDataObject != NULL) {
        if(messengerPortsArray[(*flowListIterator)->getDstPort()]) {
                if(sizeof(srcHostDataObject->getH1F5F6()) >= 10) {
                        tempBitMask = srcHostDataObject->getBitMaskOfH1F5F6OnIP((
                        *flowListIterator)->getDstIP());
                        if(tempBitMask & bitMaskArray[10] > 10){
                                tempBitMask = (*flowListIterator)->getClassmask();
                                tempBitMask = tempBitMask | bitMaskArray[10];
                                (*flowListIterator)->setClassmask(tempBitMask);
                        }
                }
        }
}
```

F6 Gaming

Aus [1] : „ ***F6: Gaming***:(J). Popular game servers (currently only the most common online games Half-Life and World of Warcraft) are identified in the same fashion as messenger servers. All traffic to and from the game servers on well known gaming
ports is classified as gaming traffic. “

Unsere Interpretation:

Die Identifikation dieser Kategorie ist derjenigen der Messenger-Kategorie sehr ähnlich. Im Unterschied zu Messenger aber ist der Gaming-Host typischerweise mit bekannten Gaming-Ports verbunden.

```
//Aggregatorphase
keine, da diese gleich ist wie H1
//Processing Phase
if(srcHostDataObject != NULL) {
        if(messengerPortsArray[(*flowListIterator)->getDstPort()]) {
                if(sizeof(srcHostDataObject->getH1F5F6()) >= 10) {
                        tempBitMask = srcHostDataObject->getBitMaskOfH1F5F6OnIP((
                        *flowListIterator)->getDstIP());
                        if(tempBitMask & bitMaskArray[10] > 10){
                                tempBitMask = (*flowListIterator)->getClassmask();
                                tempBitMask = tempBitMask | bitMaskArray[11];
                                (*flowListIterator)->setClassmask(tempBitMask);
                        }
                }
        }
}
```

F7 FTP

Aus : „ ***F7: Ftp***: (J). Ftp was not taken into account by Karagiannis, while Perenyi implicitly included it as part of its 'well known port' rule. Identifying data transfer in passive ftp remains a problem. Active ftp data transfer on the other hand can easily be marked as ftp traffic identified by an initiating sourceport number of 20, as used by ftp servers to actively serve their requesting clients. “

Unsere Interpretation:

Datenverkehr dieser Kategorie kann aufgrund der Quellportnummer 20 (Control Port) und 21 (Data Port) identifiziert werden. Auf dieser Portnummer hört der Ftp-Server und nimmt Client-Anfragen entgegen. Zu Beachten ist hier die Unterscheidung zwischen Quell und Zielport.

```
//Aggregatorphase
keine
//Processing Phase
if(ftpPortsArray[(*flowListIterator)->getSrcPort()] || ftpPortsArray[(*flowListIterator)->getDstPort()]){
        tempBitMask = (*flowListIterator)->getClassmask();
        tempBitMask = tempBitMask | bitMaskArray[12];
        (*flowListIterator)->setClassmask(tempBitMask);
}
```

F8 nonP2PPorts

Aus [1] : „ ***F8: non P2P Ports***:(P). As noted by Perenyi, destination ports are still suitable to identify traffic of some common applications. Our set of well known non-P2P ports includes netbios, dns, telnet, ssh, ftp, mail, rtp and bgp. All flows to the listed destination ports are marked as non-P2P flows. “

Unsere Interpretation:

Eine definierte Liste mit bekannten Portzuweisungen wird mit dem Ziel-Port eines Flows verglichen und entsprechend markiert. Alle Flows, die diesem Kriterium entsprechen sind non P2P Port – Flüsse.

```
//Aggregatorphase
keine
//Processing Phase
if(wkNonP2PportsArray[(*flowListIterator)->getDstPort()]) {
        tempBitMask = (*flowListIterator)->getClassmask();
        tempBitMask = tempBitMask | bitMaskArray[13];
        (*flowListIterator)->setClassmask(tempBitMask);
}
```

F9 Attacks

Aus [1] : „ ***F9: Attacks***:(J). This rule is probably the most significant improvement to the original heuristics. While Perenyi does not take malicious traffic into account at all, Karagiannis rules out simple network scans as false positives. We first identify suspicious pairs of source IPs and destination Ports (*AttackPairs*). All flows with source IP and destination port inside the list of AttackPairs are then marked as attacks. AttackPairs are identified by three different cases:
a) *Sweep*: The ratio between number of destination IPs (#dIP) and number of destination ports (#dPort) from a certain host is greater than 30. This means that one host is connecting to a lot of hosts with only a few different port numbers, as typically the case when scanning IP ranges for vulnerabilities on specific ports.
b) *Scan*: The ratio between #dIP and #dPort is less than 0.33 and #dIP is less than 5. This would be the case if one host is scanning a small number of specific, dedicated targets on a large number of different ports. c) *DoS*: #dIP is less than 5, #dPort is less than 5 and the average number of conn. per sec (conn/s) is greater than 6. This behavior represents 'hammering' attacks, where one host is trying to overload a few targets (typically one) by opening connections to a few services very frequently. "

Unsere Interpretation:

Attacken-Paare (Quell-IP und Ziel-Port) werden wie folgt unterschieden:

a.) **Durchlauf:** Wenn der Quotient zwischen der Ziel-IP – Anzahl und der Ziel-Portnummer – Anzahl grösser als 30 ist, dann ist es wahrscheinlich eine Attacke. Host versucht mit festen Ziel-Portnummern zu vielen Ziel-IP's eine Verbindung aufzubauen.
b.) Scan: Wenn der Quotient zwischen der Anzahl Ziel-IP's und der Anzahl Ziel-Portnummern kleiner als 0.33 ist, dann wird der Fluss auch als Attacke identifiziert. (Ein Host scannt eine kleinere Gruppe von Hosts mit einer grossen Anzahl von Ziel-Portnummern).
c.) DoS: Wenn die Anzahl von Ziel-IP's und die der Ziel-Portnummern kleiner als 5 und die Verbindungsversuche/Sekunde grösser als 6 ist, dann nennt man das eine „Hammering"-Attacke, in der ein Angreifer versucht den Zielhost zu überfluten.

```
//Aggregatorphase
keine, da gleich wie H4
//Processing Phase
if(srcHostDataObject != NULL) {
        if(((interval->getEndPointKeyHashMapIterator()->second.countOfConnectedIPs())/
        (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedPorts()) > 30)||
        (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedIPs())/
        (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedPorts() > 3 &&
        (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedIPs()) < 5) ||
        ((interval->getEndPointKeyHashMapIterator()->second.countOfConnectedIPs()) < 5 &&
        (interval->getEndPointKeyHashMapIterator()->second.countOfConnectedPorts()) < 5)) {
                interval->getEndPointKeyHashMapIterator()->second.setF9(true);
                tempBitMask = (*flowListIterator)->getClassmask();
                tempBitMask = tempBitMask | bitMaskArray[14];
                (*flowListIterator)->setClassmask(tempBitMask);
        }
}
```

F10 Unclassified nonP2P Port and Flows

Aus [1] : „ ***F10: unclassified, known non-P2P Port***:(J). Up to this point all heuristics mark flows independent of each other. All flows left unmarked until now are neither suspected to be P2P traffic nor obvious cases of non-P2P traffic. We believe it is safe now to apply a port number classification on the previously unclassified flows. All flows, whose source- or destination port number matches a set of well-known non-P2P port numbers including (http, messenger, game) are classified non-P2P, if not classified by any heuristics (H1-H4, F1-F9). “

Unsere Interpretation:

Die restlichen Datenflüsse, die übrig geblieben sind und nicht mit den vorherigen Heuristiken übereinstimmen werden als non-P2P markiert.

```
//Aggregatorphase
keine
//Processing Phase
tempBitMask = (*flowListIterator)->getClassmask();
if(tempBitMask & bitMaskArray[0] == 0){
        tempBitMask = tempBitMask | bitMaskArray[15];
        (*flowListIterator)->setClassmask(tempBitMask);
}
```

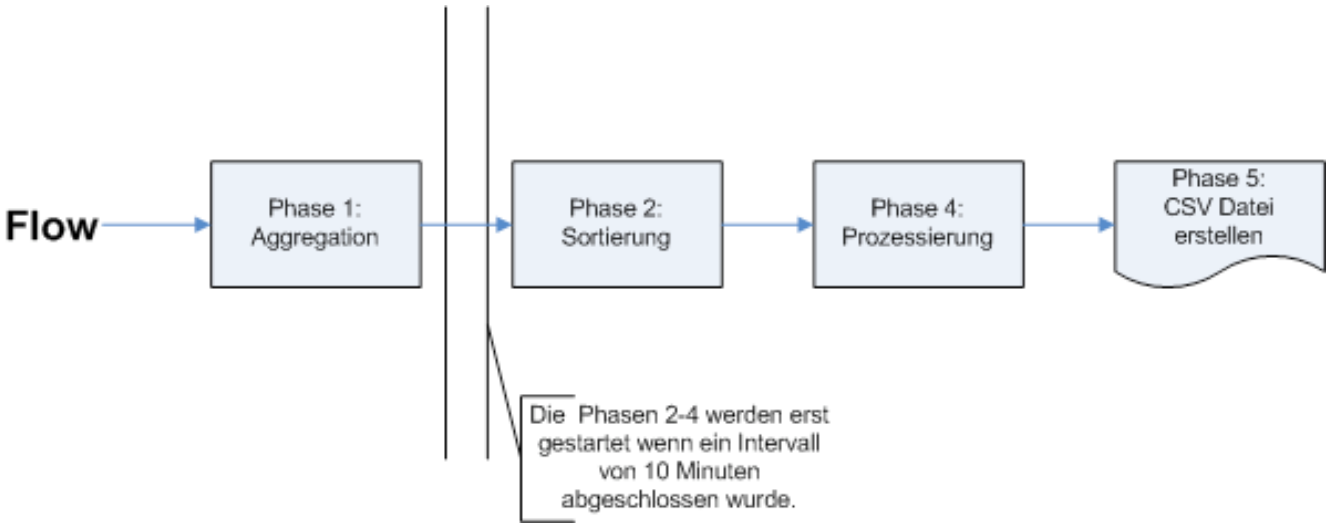
1.1 Zeit und Raumkomplexität

Es wurde zudem festgestellt, dass etwa 20 Millionen Flows in den 10-Minuten-Intervallen fliessen. Die geschätzte Anzahl eindeutiger Hosts, die in diesem Zeitrahmen jemals mit einer Gegenstation kommunizieren, ergibt sich aus dem Quotient aus der Anzahl Flows / 10. Ein so grosses Datenaufkommen in einer so kurzen Zeit verlangt entsprechend effiziente Algorithmen und Datenstrukturen. Hierfür haben wir eine Zeit- und Raumkomplexitäts-Analyse erstellt.

Aufgrund dieser Analyse und der erarbeiteten Lösungsvarianten für die verschiedenen Speichermöglichkeiten haben wir individuelle, optimal geeignete HashMaps als Datenstruktur implementiert (Key, Value-Paare), die den Anforderungen der Heuristiken gerecht werden. Aus dieser Analyse geht zudem hervor, dass auch grössere Mengen ohne Probleme in endlicher Zeit verarbeitet werden können. Die genauen Details findet man im Dokument zur Zeit- und Raumkomplexitätsanalyse.

2. Teil: Implementierung

Vom Einlesen der Daten bis hin zur Auswertung und dem Schreiben der Ergebnisdaten in das CSV-File unterscheiden wir 4 Phasen:



Phase 1: Sie beinhaltet die Aggregation der einzelnen Flows. In der Aggregationsphase werden die einkommenden Flows werden so aggregiert, dass die Processing-Phase bzw. Evaluierungsphase dies Daten anhand von verschiedenen Heuristik-Kriterien einfach abfragen und eine entsprechende Statistik schreiben kann.

Die Heuristik-Regeln verlangen sowohl eine host- als auch eine flowspezifische Verarbeitung von Daten. Aus diesem Grund haben wir eine zentrale HashMap namens HostHashMap erstellt. Sie speichert sogenannte Host-Objekte (HostDataObject). Mittels IP (Source-IP) kann man mit einem konstanten Laufzeitverhalten von O(1) einen Eintrag schnell suchen, worauf man als Resultat das Host-Objekt bekommt. Ein Host-Objekt enthält alle host- und heuristikspezifischen Informationen. Je nach Heuristik haben wir im Host-Objekt (HostDataObject) zusätzliche HashMaps definiert, die Daten speichern, um sie in der Processing-Phase nach Ablauf der 10 Minuten evaluieren zu können. Da die Flows in dieser Phase noch nicht sortiert sind, werden sie mittels eines Modulo-Algorithmus den einzelnen Intervallen zugeordnet, indem die Startzeit der einzelnen Flows ausgewertet wird.

Während des Durchlaufs sind immer 3 Intervalle aktiv. Da die Aufgabenstellung eine intervallspezifische Speicherung und Verarbeitung der Flow-Daten erfordert, definieren wir Intervall-Objekte. Jedes dieser 3 aktiven Intervall-Objekte hat seine eigene Liste mit den dazugehörigen Flows in Form von 8-Tupel-Informationen (Source-IP/-Port, Destination-IP/-Port, Protokoll, ToS, Start-Zeit, Klassenmaske), welche bei jedem Schleifendurchgang durch die Flow-Liste mit dem aktuellen Flow ergänzt wird. Das Intervall-Objekt speichert auch die oben erwähnte HostHashMap mit den Host-Objekten als Inhalt und eine Endpunkt-HashMap (EndPointKeyHashMap), welches Endpunktwert-Objekte (EndPointValue) als Wert und einen Endpunktschlüssel (EndPointKey) als Schlüssel speichert. Ein Endpunktwert-Objekt (EntPointValue gibt Auskunft über die Anzahl eindeutiger Destination-IP's und –Ports, die der entsprechende Endpunkt jemals in den 10 Minuten kontaktiert hat. Sobald ein Flow in den 3. Intervall kommt, wird der erste abgeschlossen. Die ersten Intervall-Daten gehen dann in die 2. Phase (Sorting), anschliessend in die 3. Phase (Processing) und schlussendlich in die 4. Phase (CSV schreiben) über.

Da es auch vorkommt, dass während einer Zeitspanne aufgrund einer technischen Panne keine Daten hereinkommen, ist das Zeitfenster-Handling so implementiert, dass das zu verarbeitende Zeitfenster immer bis zum aktuellen Flow nachgerückt wird und der abgeschlossene Intervall (2. - 4. Phase) sofort verarbeitet wird. Mittels dieser Intervall-Aufteilung und des Modulo-Algorithmus können diese Zeitlücken bewältigt werden. Somit ist eine dynamische Intervallverarbeitung gegeben.

Phase 2: Sie beinhaltet die Sortierung der Flows. Da bei einzelnen Heuristiken die Zeit eine grosse Rolle spielt, werden die Flows noch vor der Auswertung nach dem Zeitkriterium sortiert.

Phase 3: Sie ist die Processing-Phase. In dieser Phase wird durch die abgefüllte Flow-Liste des abzuschliesenden Intervall-Objektes iteriert und mit Hilfe bestimmter Heuristik-Kriterien ausgewertet. Die Markierung einzelner Klassen erfolgt über eine Bitmaske. Jedes einzelne Bit dieser Maske stellt eine Applikationsklasse dar. Die Maske wird immer der Heuristik-Regel entsprechend gesetzt. H1 = 1. Bit/ H2 = 2. Bit/ usw. Jedem Flow-Record, der durch das Modul abgearbeitet wird, wird also eine Bitmaske zugewiesen (siehe 8-Tupel-Information). Anhand dieser Bitmaske kann dann evaluiert werden, welche Regeln eingetreten sind und welche nicht. Jede Regel hat ihren eigenen Zähler, der bei jedem Regeltreffer kontinuierlich inkrementiert wird.

Phase 4: Sie beinhaltet das Schreiben des CSV-Files. Hier werden die Zähler der einzelnen Heuristik-Regeln ausgewertet und in das CSV-File geschrieben. Jedes Intervall bekommt eine Zeile mit dem Zustand der einzelnen Zähler pro Applikationsklasse.

2.1 Einstellungen

2.1.1 Installationseinstellungen

Das SIAN-Modul ist eine C++ Applikation mit einem Makefile. Zusätzlich braucht das SIAN-Modul noch die kompilierte Library des Frameworks.

Das Modul besitzt zudem noch eine eigene Main-Klasse, sodass es unabhängig vom Framework gestartet werden kann; es braucht aber die Library des Frameworks. Um das SIAN-Modul zu starten, braucht man das Makefile, um eine Bin-Datei zu generieren. Diese Bin-Datei ruft man dann mit den Flows und der Start- und Stopzeit als Parameter auf.

2.1.2 Liste der Codedateien

Dateiname	Pfad
Aggregator.h	sian_src/include/
HashKeyIPv4_2T.h	sian_src/include/
HostDataObject.h	sian_src/include/
Interval.h	sian_src/include/
IPv4_8T.h	sian_src/include/
M_sian_prototyp.h	sian_src/include/
Processer.h	sian_src/include/
Sorter.h	sian_src/include/
To_Console.h	sian_src/include/
Aggregator.cpp	sian_src/src/
HashKeyIPv4_2T.cpp	sian_src/src/
HostDataObject.cpp	sian_src/src/
Interval.cpp	sian_src/src/
IPv4_8T.cpp	sian_src/src/
M_sian_prototyp.cpp	sian_src/isrc/
Processer.cpp	sian_src/src/
Sorter.cpp	sian_src/src/
To_Console.cpp	sian_src/src/

2.1.3 Allgemeine Einstellungen

Das SIAN-Modul wurde so entwickelt, dass die Intervalle im 10-Minuten-Takt erfolgen. Diese Einstellung lässt sich aber ohne grösseren Codeaufwand ändern. Auch können die Heuristik-Regeln erweitert werden. Da in diesem Fall ein uint32_t mit 32 Bits verwendet wird, können die Regeln bis auf insgesamt 32 ergänzt werden. Die Start- und Stop-Zeit, die für die Intervalle benutzt wird, können bequem über die Kommandozeile beim Modulaufruf mitgegeben werden.

Das SIAN-Modul verwirft zur Zeit noch IPv6-Pakete und wertet lediglich IPv4-Pakete aus. Die Erweitertung auf IPv6-Pakete wäre aber sicherlich denkbar und nützlich.

2.2 Testen

Nachdem wir uns einige Gedanken darüber gemacht haben, wie man einen effizienten Test schreibt, um das SIAN-Modul zu testen, sind wir auf die Idee gekommen, die Heuristik-Regeln mit vorpräparierten Daten, welche einen Regel-Ausschlag provozieren, gezielt zu testen. Jede Regel generiert einen spezifischen Output (z.B. H1 TCP/ UDP IP Paare, 1. Bit in Klassenmaske und somit einen Output gemäss Bitmaske = 000000000000001). Somit kann man pro Regel-Test mittels der "DIFF"-Funktion einen erwarteten mit einem effektiven Text vergleichen und feststellen die einzelnen Regel-Implementationen richtig funktionieren oder nicht. Unsere Devise lautete also:

"Blackbox-Test statt Unit-Test".

Es wurde ganz bewusst auf Unit-Test verzichtet. Ein Grund war sicherlich der Mangel an Zeit. Ein zweiter Grund war, dass mittels des oben erwähnten Blackbox-Tests die Funktionalität der einzelnen Heuristik-Regeln getestet werden konnte.

Der Test betreffend der Zeit- und Raumkomplexität (bzw. mit realem Datenverkehr) konnte wegen Mangel an Daten und Zeit nicht durchgeführt werden. Dieser kann aber nach der Abgabe der Semesterarbeit immer noch nachgeholt werden und die nötigen Anpassungen können ohne Weiteres vorgenommen werden.

5.3. Schlussfolgerungen

Was wurde erreicht?

Unsere Semesterarbeit hat das bestehende Framework um das Modul SIAN erweitert. Das SIAN-Modul analysiert während der Zeit, welche durch die vom User eingegebene Start und Stopp-Zeit definiert wird, die kommenden Flow-Records und wertet diese nach den verschiedenen Applikationsklassen aus. Zur Identifikation der Applikationsklassen haben wir die entsprechenden Heuristiken aus der Literatur analysiert und an das Framework angepasst.

Die Flows werden in 10-Minuten-Intervalle aufgeteilt. Diese Einstellung lässt sich aber ohne grossen Aufwand im Code ändern. Die Auswertungsdaten der einzelnen Intervalle werden in ein CSV-File geschrieben. Diese Datei kann dann für weitere Forschungszwecke genutzt werden.

Vergleich mit anderen Lösungen

Heutzutage gibt es schon Ansätze von anderen Universitäten und Hochschulen, die sich mit diesen Heuristiken befasst haben. Deren Erkenntnisse konnten wir uns zu Nutze machen. Doch gab es keine Anwendung, die unseren Anforderungen entsprachen. Aus diesem Grund haben wir in unserer Semesterarbeit das Framework mit dem Modul SIAN erweitert, das die Heuristiken nach den Anforderungen des Frameworks implementiert hat.

Was wurde nicht erreicht?

Das Modul ist auf theoretischen Ebene fertiggestellt, bei dieser Semesterarbeit fehlten uns die Zeit und die erforderlichen Daten, um die Zeit- und Raumkomplexität zu testen.

Was müsste noch getan werden?

Momentan werden nur IPv4 Pakete ausgewertet und IPv6-Pakete werden verworfen. Die Erweiterung auf IPv6-Pakete wäre sicherlich nützlich und wird in nächster Zeit wahrscheinlich auch nötig sein. Darüber hinaus müssten die Tests betreffend die Zeit- und Raumkomplexität durchgeführt werden.

Weiteres Vorgehen

Da es sich hier um eine Applikation mit konkreter Anwendung handelt, müssen sicherlich die Tests betreffend Zeit- und Raumkomplexität nachgeführt werden. Auch die Erweiterung auf IPv6 muss in absehbarer Zeit implementiert werden.

Die Übergabe der Intervallgrösse per Kommandozeile wäre sicherlich ein nützliches Feature.

Eine Möglichkeit bestünde darin, dass wir nach der Semesterarbeit als Praktikanten an diesem Projekt weiterarbeiten und es möglicherweise im Frühlingssemester 2010 als Bachelorarbeit beenden. Diese Punkte müssen aber noch mit den Projektbetreuern abgeklärt werden.

Bedeutung des Ergebnisses

Das Framework wird für Forschungszwecke an der ETH genutzt. Das SIAN Modul ist ein Teil dieses Frameworks, das zur Identifikation der Applikationsklassen dient. In Zukunft wird man dank der Erweiterung des Frameworks mit dem SIAN Modul Netzwerkverkehrsdaten genauer analysieren und feststellen können, um welche Applikationsklassen es sich handelt. Die Identifikation der Applikationsklassen könnte auch für den Provider, welcher die Daten zur Verfügung stellt, von Nutzen sein.

6. Persönliche Bereichte

6.1. Persönlicher Bericht von Selim Akyol

Projektverlauf

Wir hatten den Vorteil, dass dieses Projekt gleichzeitig auch ein reales Projekt mit konkreter Anwendung war. Der Realitätsbezug hat die Motivation gesteigert. Das SIAN-Projekt sprengt den Rahmen einer Semesterarbeit und wird höchstwahrscheinlich noch weiter gehen, da schon zusätzliche Wünsche und Erweiterungsmöglichkeiten geäussert wurden und Erweiterungsmöglichkeiten bestehen. Das Team, in welchem ich eingebunden bin, gefällt mir sehr. Das Arbeiten mit meinem Teamkollegen und unseren Projektbetreuern macht Spass.

Was habe ich gelernt?

Eine wichtige Erfahrung für mich war, dass bei grösseren Projekten ohne genaue Planung nichts funktioniert. Zwar habe ich die Planung zu Beginn des Projekts als mühselig und nicht wirklich wichtig empfunden, doch ziemlich schnell musste ich einsehen, dass sie unabdingbar und extrem wichtig ist.

Ein anderes Problem war, dass wir uns wieder in die Programmiersprache C++ einarbeiten mussten. Auch mussten wir einen Einblick darüber bekommen, wie das aktuelle Framework aussieht und wie es funktioniert. Dazu mussten wir uns noch mit dem Format NetFlow von Cisco vertraut machen.

Die Analyse der verschiedenen Heuristiken und deren Anpassung an das Projekt waren recht komplex. Es brauchte einige Anstrengungen bis jede einzelne Regel auf den Beinen stand. Die anschliessende Analyse und Suche von geeigneten Algorithmen und entsprechenden Datentypen gehörte von mir aus gesehen zu den spannendsten Teilen dieses Projekts. Wir mussten wir für jede einzelne Regel schauen, welche Informationen in welchem Zusammenhang gespeichert werden sollen.

Fazit

Das Projekt SIAN ist ein Modul eines Frameworks, das zu Forschungszwecken genutzt wird. Wir haben das Modul so weit entwickelt, dass es eine gute Basis darstellt. Allerdings ist es noch ein Prototyp und braucht eine weitere Bearbeitung bis es allenfalls auch von Dritten wie z.B. Switch genutzt werden kann.

6.2. Persönlicher Bericht von Edon Berisha

Projektverlauf

Da ich wusste, dass dieses Projekt nach Abschluss für Forschungszwecke effektiv Anwendung finden würde, war meine Motivation schon von Beginn an gross. Dank einem eingerichteten, zentralen SVN-Repository, wo wir unsere Code- und Dokumentationsdateien gespeichert haben, lief die Projektkoordination bezüglich das Abändern und Hinzufügen von Daten (Code, Dokumente, etc.) ohne nennenswerte Arbeitskollisionen.

In den wöchentlichen Teammeetings konnten wir immer den aktuellen Zwischenstand besprechen, ausgearbeitete Ergebnisse diskutieren und einen Plan für das weitere Vorgehen erstellen. Durch die gute Organisation und Koordination des Projekts erhielten wir immer die aktuellen Aufgaben zugeteilt und konnten so die nächsten Schritte besprechen. Die Kommunikation zwischen den Projektbeteiligten (Projektmitglieder, Projektleiter und Auftraggeber) funktionierte wunderbar und auf die Unterstützung des Projektleiters und die des Auftraggebers konnten wir immer zählen.

Was habe ich gelernt?

Den für dieses Projekt notwendigen Dokumentationsaufwand habe ich extrem unterschätzt. Also empfehle ich dringend eine fortlaufende Pflege der jeweiligen Dokumentationen und werde dieses Anraten für das nächste Mal sicherlich beachten.

In der Entwicklung dieses Projektes habe ich mein Wissen über C++ extrem vertieft. Obwohl ich schon vorher Grundkenntnisse über C++ hatte, habe ich viel Neues dazugelernt.

Die grösste Sorge am Anfang des Projekts war der Zeit- und Speicherverbrauch für die Verarbeitung so grosser Datenmengen. Auch die Suche nach den geeigneten Datenstrukturen und Algorithmen bereitete uns zuerst Sorgen. Doch nach einigen Recherchen im Internet und nach einigen Meetings mit den Projektbetreuern haben wir eine Lösung gefunden die eine Auswertung in endlicher Zeit erlaubt. Die Bewältigung dieser Hürde war sicherlich auch eine grosse Lehre – auch für die Zukunft.

Fazit

Das Ziel des Projekts für diese Semesterarbeit war die Kernfunktionalität zu implementierten, was auch erfolgreich erreicht wurde. Es war grundsätzlich ein sehr lehrreiches Projekt mit vielen Hürden und Freuden. Für zukünftige Projekte werde ich auf jeden Fall die Dokumentationen fortlaufend ergänzen. Zur Erleichterung der Projektkoordination zwischen den Projektmitgliedern würde ich das nächste Mal allenfalls geeignete open-source-Tools einsetzen.

7. Glossar

Begriff	***Beschreibung***
NetFlow	Netflow ist eine Technik, bei der ein Gerät, in der Regel ein Router oder Layer-3-Switch, Informationen über den IP-Datenstrom in das Gerät per UDP exportiert. Diese UDP-Datagramme werden von einem Netflow-Kollektor empfangen, gespeichert und verarbeitet. Die anfallenden Daten werden für Verkehrsanalysen, für Kapazitätsplanungen oder QoS-Analysen verwendet.
Heuristik	Heuristik bedeutet: Mit geringem Rechenaufwand und kurzer Laufzeit zulässige Lösungen für ein bestimmtes Problem zu erhalten. Klassische Algorithmen versuchen, einerseits die optimale Rechenzeit und andererseits die optimale Lösung zu garantieren. In unserem Fall geht es darum, mittels geeigneten Verfahren die verschiedenen Applikationsklassen zu identifizieren.
Applikationsklasse	In unserem Fall werden nach P2P- und NonP2P Klassen unterschieden. Die NonP2P Klassen werden in 9 weitere Klassen, wie z.B. Web, FTP, DNS etc. unterteilt werden.
Flow	Ein Datenfluss. Von Source- nach Destination-Gerät.
H1 TCP/UDP Pair	Quell- und Destination-IP – Paare, die nebenläufig TCP und UDP benutzen werden deshalb als P2P-Verkehr markiert. Mit nebenläufig ist hier gemeint, dass in einem 10 Minuten – Intervall TCP- und UDP-Pakete vorkommen.
H2 P2P Ports	Verkehrsflüsse, die typische P2P Ports benutzen, werden als P2P-Fluss markiert.
H3 Port Benutzung	Wenn der gleiche Quellport wiederholend auf einem Host innerhalb von 60 Sekunden gebraucht wird, so ist es ein P2P-Host und alle Flüsse von und zu diesem Host sind P2P-Flüsse.
H4 P2P Pair	Wenn also die Anzahl eindeutiger Quell-Port-Nummern – Quell-IP's < 2 ist und die Anzahl eindeutiger Quell-IP's > 5, dann wird dieser Host als P2P-Host vermerkt und alle Flüsse von und zu diesem als P2P-Fluss markiert.
H5	Datenflüsse, die nicht als H1, H2, H3 oder H4 markiert wurde und eine Grösse von über 1 MB haben und eine Dauer von mehr als 10 Min aufweisen, werden als H5 markiert.
F1 Web Pair	Alle Flüsse mit den Portnummern 80, 443 und 8080 zu und von diesem Webhost sind als Webflüsse markiert.
F2 Web	Hosts (Clients) mit parallelen Verbindungen zu einem Http-Port werden als Webhost markiert und alle Flüsse von und zu einem Webserver als Webflüsse.
F3 DNS	Wir markieren Endpunkte (IP, Port) als non-P2P (DNS), wenn sie Flüsse mit gleichen Quell- und Zielportnummern, die kleiner als 501 sind, beinhalten.
F4 Mail	Hosts, die Datenverkehr am Mailport empfangen und neue Verbindungen zu anderen Hosts mit der Portnummer 25 starten markieren wir als Mail-Server.
F5 Messenger	Hosts, die mit mindestens 10 verschiedenen IP's auf bekannten Messenger-Ports kommunizieren, markieren wir als Messenger-Host.
F6 Gaming	Hosts, die mit mindestens 10 verschiedenen IP's auf bekannten Gaming-Ports kommunizieren, markieren wir als Gaming-Host.
F7 FTP	Datenverkehr dieser Kategorie kann aufgrund der Quellportnummer 20 (Control Port) und 21 (Data Port) identifiziert werden.
F8 nonP2PPorts	Eine definierte Liste mit bekannten Portzuweisungen wird mit dem Ziel-Port eines Flows verglichen und entsprechend markiert.
F9 Attacks	Aufgrund von Attackenpaaren werden Attacken identifiziert. Dabei werden 3 Kategorien unterschieden: - Sweep (Kontaktieren von wenigen Ports auf vielen Hosts).

	- Scan (Kontaktieren von vielen Ports auf wenigen Hosts). - DOS (Kontaktieren von weniger als 5 Hosts).
F10 Unclassfied	Die restlichen Datenflüsse, die übrig geblieben sind und nicht mit den vorherigen Heuristiken übereinstimmen werden als unclassified flows markiert.

8. Literaturverzeichnis

Bezeichnung	Verzeichnis
NetFlow	file:///home/betellen/csg_cluster_svn/netflow/NetflowVX/NetflowVxPlusPlus/doc/html/index.html
Portliste	sian_docs/docs/Unterlagen/portliste.pdf
ConPattern	sian_docs/docs/Unterlagen/ConPattern-Based P2P Application.pdf
Guide 64 Bit	sian_docs/docs/Unterlagen/guide_to_64bit.pdf
Heuristics	sian_docs/docs/Unterlagen/Heuristic_to_Classify_Internet_Backbone_Traffic_based_on_Connection_Patterns_ICOIN2008.pdf
Analysis of P2P	sian_docs/docs/Unterlagen/Identification_and_Analysis_of_Peer-to-Peer_Traffic_2006.pdf
TCP / IP	sian_docs/docs/Unterlagen/tcpip.pdf
TCP Identification	sian_docs/docs/Unterlagen/Transport_Layer_Identification_of_P2P_Traffic_2004.pdf
C++.com	www.cplusplus.com

9. Ressourcen

[1] John Wolfgang and Sven Tafvelin, ***Heuristics to Classify Internet Backbone Traffic based on Connection Patterns***, Departement of Computer Sience and Engieneering Chalmers University of Technology, Göteborg, Sweden

[2] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, ***Transport Layer Identification of P2P Traffic***, UC Riverside, CAIDA, SDSC

[3] Marcell Perényi, Trang Dinh Dang, Adnràs Gefferth, Sàndor Molnàr, ***Identification and Analysis of Peer-to-Peer Traffic***, Budapest University of Technology & Economics, Departement of Telecommunications & Media Informatics, Budapest, Hungary

10. Dokumente des Projekts

10.1. Anforderungsspezifikation

10.1.1. Einführung (Introduction)

10.1.1.1. Zweck

Dieses Dokument beinhaltet die Übersicht über die Planung und Organisation des Projektes. Es werden die Management Abläufe und Risiken, sowie die Qualitätsmassnahmen beschrieben.

10.1.1.2. Gültigkeitsbereich

Dieses Dokument ist für die gesamte Projektzeit gültig. Änderungen bleiben vorbehalten.

10.1.1.3. Definitionen und Abkürzungen

Das Glossar mit den Definitionen und Abkürzungen wird während des Projektes fortlaufend erweitert.

Siehe „Glossar.pdf“

10.1.1.4. Referenzen

- Anforderungsspezifikationskatalog „Anforderungsspezifikation.pdf“
- Notizen und Unterlagen der Sitzungen im Ordner „Unterlagen“
- Literaturverzeichnis: Siehe „http://cvsi.hsr.ch/svn/SIAN“

10.1.2. Allgemeine Beschreibung

10.1.2.1. Projekt Übersicht

An der ETH Zürich werden zu Forschungszwecken Datenanalysen an realen Verkehrsdaten eines mittelgrossen schweizerischen Internet Service Providers (ISP) durchgeführt. Dazu steht eine grosse Datensammlung (komprimiert ca. 29 TB) von CISCO NetFlow-Daten zur Verfügung, die fortlaufend um die neuesten Verkehrsinformationen ergänzt werden. Um diese Daten zu verarbeiten stehen ein leistungsfähiges Software Framework sowie ein moderner Computer Cluster zur Verfügung.

10.1.2.2. Produkt Funktion

Das Ziel dieser Arbeit besteht darin, das bestehende Framework um ein Modul zu ergänzen, das die Identifikation verschiedener Applikationsklassen unterstützt. Zur Umsetzung der entsprechenden aus der Literatur bekannten Heuristiken ist die Wahl effizienter Datenstrukturen und Algorithmen eminent wichtig, da nur so die grossen Datenmengen in endlicher Zeit verarbeitet werden können. Dazu sind die Heuristiken zuerst zu analysieren, bevor die Erstellung der Software in C++ in Angriff genommen wird.

10.1.2.3. Benutzer Charakteristik

Mitarbeiter der ETH:	Integrierung des Moduls in das bestehende Framework zur Analyse des Internettraffics. Benutzung des SIAN-Moduls als einzelnen oder in Verbindung mit dem ganzen Framework.
Switch:	Nutzung des SIAN-Moduls und des bestehenden Frameworks für statistische Zwecke.

10.1.2.4. Einschränkungen und Annahmen

Wir gehen von folgenden Annahmen und Einschränkungen aus:

- Das Einarbeiten in das bestehende Framework dauert nicht länger als 3 Wochen.
- Die Schnittstellen zu den anderen Modulen sind verfügbar und dokumentiert.
- Die Richtzeit pro Mitglied wird auf 16 Stunden die Woche angenommen.

10.1.2.5. Abhängigkeiten (Dependenencies)

Das SIAN-Modul ist abhängig von einzelnen Modulen des bestehenden Frameworks. Vor allem die Module, die NetFlow-Daten holen und diese für die Analyse bereitstellen.

10.1.3. Spezifische Anforderungen

10.1.3.1. Funktionale Anforderungen

Das zu entwickelnde Modul identifiziert nach dem Einlesen der NetFlow-Dateien die verschiedenen Applikationsklassen und stellt das Resultat in Form einer Zusammenfassung zur Verfügung.

Um das Netzwerkdesign ständig zu verbessern und das „Quality of Service“ unterstützen zu können müssen die Netzwerkbetreiber die Verkehrstypen bzw. Applikationsklassen identifizieren können.

Die Heuristiken werden nach deren Autoren wie folgt unterteilt:

(K) Karagiannis, (P) Perenyi, (J) John

Die Reihenfolge der Regelanwendung ist identisch mit der nachfolgenden Aufzählungsreihenfolge.

Bei der Paketanalyse wird wie folgt aufgrund von Verbindungseigenschaften unter folgenden Applikationsklassen unterschieden:

10.1.3.1.1. Peer-To-Peer

Karagiannis und Perenyi haben verschiedene Heuristiken definiert um ein P2P-Flow zu identifizieren:

10.1.3.1.2. H1: TCP/UDP IP Paare (K), (P)

Viele Peer-To-Peer Applikationen nützen das TCP-Protokoll um Daten auszutauschen und das UDP-Protokoll um einen Datenverkehr zu signalisieren. Quell- und Ziel-IP – Paare, die nebenläufig TCP und UDP benutzen werden deshalb als P2P-Verkehr markiert. Mit nebenläufig ist hier gemeint, dass in einem 10 Minuten – Intervall TCP- und UDP-Pakete vorkommen.

10.1.3.1.3. H2: P2P Ports (P)

Obwohl viele P2P-Applikationen heutzutage beliebige Ports verwenden, können dennoch ca. 1/3 vom gesamten P2P-Datenverkehr aufgrund der typischen P2P-Zielports identifiziert werden. Verkehrsflüsse, die diese Zielports verwenden, werden als P2P-Fluss markiert.

10.1.3.1.4. H3: Port Benutzung (P)

In normalen Anwendungen werden kurzlebige Quellports vom Betriebssystem bestimmt. Deshalb ist es sehr ungewöhnlich, dass gleiche Quellports innerhalb von kurzen Zeitperioden verwendet werden. Wenn der gleiche Quellport wiederholend auf einem Host innerhalb von 60 Sekunden gebraucht wird, so ist es ein P2P-Host und alle Flüsse von und zu diesem Host sind P2P-Flüsse.

10.1.3.1.5. H4: P2P IP/Port Paare (K)

P2P-Peers halten normalerweise nur eine einzelne offene Verbindung zu einem anderen Peer offen, was heisst, dass jeder Endpunkt (IP, Port) eine gleich grosse IP-Liste mit eindeutigen Quell-IP-Adressen und eindeutigen Quell-Ports beinhaltet. Wenn also die Anzahl eindeutiger Quell-Port-Nummern – Quell-IP’s < 2 ist und die Anzahl eindeutigerQuell-IP’s > 5, dann wird dieser Host als P2P-Host vermerkt und alle Flüsse von und zu diesem als P2P-Fluss markiert.

10.1.3.1.6. H5: Undefined P2P (K)

Datenflüsse die nicht als H1, H2, H3 oder H4 markiert wurde und eine Grösse von über 1MB haben und eine Dauer von mehr als 10 Min haben werden als H5 markiert.

10.1.3.1.7. F1: Web IP/Port Paare (K)

Im Gegensatz zu P2P führt ein Webserver mehrere Verbindungen. Wenn die Differenz zwischen der Anzahl eindeutiger Portnummern und eindeutiger IP-Nummern, die zu einem Endpunkt verbunden sind, grösser als 10 ist, der Quotient von Eindeutiger Portnummern/ Eindeutige IP-Nummern grösser als 2 und mindestens 10 verschiedene IP’s zu diesem Endpunkt verbunden sind (**#sPort-#sIP>10 && #sPort/#sIP && #sIP>10**), dann wird dieser Endpunkt als Webhost markiert. Alle Flüsse mit den Portnummern 80, 443 und 8080 zu und von diesem Webhost sind als Webflüsse markiert.

10.1.3.1.8. F2: Web (P)

Webclients benutzen normalerweise nicht nur mehrere, sondern sogar parallele Verbindungen zu einem Webserver. Hosts (Clients) mit parallelen Verbindungen zu einem Http-Port werden als Webhost markiert und alle Flüsse von und zu einem Webserver als Webflüsse.

10.1.3.1.9. F3: DNS (K)

DNS benutzt manchmal die gleiche Quell- und Zielportnummer. Wie Karagiannis vorschlägt markieren wir Endpunkte (IP, Port) als non-P2P (DNS), wenn sie Flüsse mit gleichen Quell- und Zielportnummern, die kleiner als 501 sind, beinhalten.

10.1.3.1.10. F4: Mail (K)

Hosts, die Datenverkehr am Mailport empfangen und neue Verbindungen zu anderen Hosts mit der Portnummer 25 starten markieren wir als Mailserver. Alle Flüsse von und zu diesem Mailserver sind Mailflüsse.

10.1.3.1.11. F5: Messenger (K)

Hosts, die mit mindestens 10 verschiedenen IP's und bekannten Messenger-Ports in einer Mindestzeitperiode von 10 Tagen verbunden sind, markieren wir als Messenger-Server. Flüsse von und zu diesem Server sind Messenger-Flüsse. Wir vermerken hier, dass wir die Überprüfung von 10 Tagen nicht machen können.

10.1.3.1.12. F6: Gaming (J)

Die Identifikation dieser Kategorie ist derjenigen der Messenger-Kategorie sehr ähnlich. Im Unterschied zu Messenger aber ist der Gaming-Host typischerweise mit bekannten Gaming-Ports verbunden.

10.1.3.1.13. F7: FTP (J)

Datenverkehr dieser Kategorie kann aufgrund der Quellportnummer 20 (Control Port) und 21 (Data Port) identifiziert werden. Auf dieser Portnummer hört der Ftp-Server und nimmt Client-Anfragen entgegen. Zu Beachten ist hier die Unterscheidung zwischen Quell und Zielport.

10.1.3.1.14. F8: non P2P Ports (P)

Eine definierte Liste mit bekannten Portzuweisungen wird mit dem Ziel-Port eines Flows verglichen und entsprechend markiert. Alle Flows, die diesem Kriterium entsprechen sind non P2P Port – Flüsse.

10.1.3.1.15. F9: Attacks (J)

Aufgrund von Attackenpaaren werden Attacken identifiziert.

Attacken-Paare (Quell-IP und Ziel-Port) werden wie folgt unterschieden:

d.) **Durchlauf:** Wenn der Quotient zwischen der Ziel-IP – Anzahl und der Ziel-Portnummer – Anzahl grösser als 30 ist, dann ist es wahrscheinlich eine Attacke. Host versucht mit festen Ziel-Portnummern zu vielen Ziel-IP's eine Verbindung aufzubauen.

e.) Scan: Wenn der Quotient zwischen der Anzahl Ziel-IP's und der Anzahl Ziel-Portnummern kleiner als 0.33 ist, dann wird der Fluss auch als Attacke identifiziert. (Ein Host scannt eine kleinere Gruppe von Hosts mit einer grossen Anzahl von Ziel-Portnummern).

f.) DoS: Wenn die Anzahl von Ziel-IP's und die der Ziel-Portnummern kleiner als 5 und die Verbindungsversuche/Sekunde grösser als 6 ist, dann nennt man das eine „Hammering"-Attacke, in der ein Angreifer versucht den Zielhost zu überfluten.

10.1.3.1.16. F10: Unclassified non-P2P Port and Flows (J)

Die restlichen Datenflüsse, die übrig geblieben sind und nicht mit den vorherigen Heuristiken übereinstimmen werden als non-P2P markiert.

Die Idee ist, dass jeder Flow alle Regelprüfungen (H1-H5 und F1-F10) durchläuft und entsprechend zugewiesen bzw. evaluiert wird. Jeder Flow kann ein P2P-Flow sein. Und um „falsche positive Aussagen" auszuschliessen gibt es die F-Heuristiken um diese „falschen positiven Aussagen" herauszufiltern und entsprechend zu kategorisieren (Web, DNS, Mail, etc.).

1. Trifft eine H1-H5 Regel zu und eine F1-F10, dann ist es die entsprechende F-Zuweisung.
2. Trifft eine H1-H4 Regel zu und keine F1-F10, dann ist es ein P2P-Flow.

10.1.4. Bedienbarkeit

10.1.4.1. Hardware

Das Modul wird im Framework auf den Cluster verwendet. Damit die grossen Datenmengen (ca. 40TB) in angemessener Zeit analysiert werden können, braucht es eine Performance von mind. 2.2 GHz Dual-Core, 16GB Ram und 2.7TiB Speicher, die vom Cluster zur Verfügung gestellt wird.

10.1.4.2. Software

Das Modul-SIAN besitzt keine Benutzeroberfläche, sondern wird im bestehenden Framework in Interaktion mit anderen Modulen genutzt.

10.1.4.3. Feedback des Betreuers

Die Applikation wird nach der Entwicklung des ersten Prototyps laufend vom Betreuer getestet und beurteilt, damit wir die Applikation nach seinen Wünschen optimal ausrichten können.

10.1.4.4. Präsentieren, nicht fragen

Das Modul wird so implementiert, dass keine Sackgassen entstehen können. Der Benutzer weiss ständig wo er steht und wie es weitergeht.

10.1.4.5. Lückenlose Rückmeldungen

Wenn ein UseCase im Gange ist, wird der Benutzer immer informiert. Damit ist eine Kommunikation zwischen Benutzer und Anwendung gewährleistet. D.h. der am Bildschirm sichtbare Zustand der Anwendung entspricht immer dem Zustand des Modells.

10.1.4.6. Nützliche Fehlermeldungen

Fehlermeldungen enthalten immer die Information:

- „Was ist passiert?“
- „Was kann der Benutzer jetzt machen?“.

Zuverlässigkeit

10.1.4.7. Verfügbarkeit

Die Verfügbarkeit des Systems soll 99% betragen. Bei Ausfall des SIAN-Moduls, sollen die anderen Module des Frameworks weiterarbeiten können.

10.1.5. Leistung (Performance)

10.1.5.1. Antwortzeit des Systems

Für die Bearbeitung eines 10minuten Intervalls sollte max. 10 Minuten Bearbeitung benötigt werden. BestCase wäre eine Realtime Verarbeitung. WorstCase sollte nach Abschluss des 2. Intervalls also nach 20Minuten die Verarbeitung abgeschlossen werden.

10.1.5.2. Speichervermögen

Auf dem Linux-Cluster stehen uns 4 GB Ram zur Verfügung und etwa 20Terabyte Speicher.

10.1.6. Wartbarkeit (Supportability)

10.1.6.1. Wartbarkeit

Die Datenbank mit den NetFlow-Daten soll dafür ausgelegt werden, dass immer grössere Datenmengen verwaltet werden können. Das SIAN-Modul muss auch mit grösseren Datenmengen (grösser als 40TB) arbeiten können.

10.1.6.2. Anpassbarkeit

Das Modul SIAN wurde in C++ geschrieben und übernimmt einige Komponente des Frameworks. Alle Anpassungen die für das Framework vorgenommen werden können also auch für das SIAN Modul übernommen werden. Mittels Codestandard wurden alle wichtigen Klassen und Methoden des SIAN-Moduls dokumentiert, sodass Erweiterungen ohne grössere Aufwände getätigt werden können.

10.1.7. Schnittstellen

10.1.7.1. Benutzerschnittstelle

Der Benutzer kann mittels Kommandozeile mit dem Framework und das SIAN-Modul kommunizieren

10.1.7.2. Hardwareschnittstelle

Per SSH wird die Verbindung zum Cluster aufgebaut.

Die Daten und das Framework befinden sich auf dem Cluster und dürfen ausschliesslich auf dem Cluster ausgeführt werden.

Die Adresse des Cluster lautet: kom-pc-aw5.ethz.ch

Der Verbindungsbefehl lautet: ssh -X kom-pc-aw5.ethz.ch -v

10.1.7.3. Softwareschnittstellen

Das Framework stellt den einzelnen Module geeignete Schnittstellen für eine Kommunikation zur Verfügung, die auch vom SIAN-Modul genutzt wird.

10.1.7.4. Datenbankschnittstelle

Das Framework besitzt schon eine Schnittstelle zur Datenbank, wo die NetFlow-Daten gespeichert sind. Ein entsprechendes Modul liest die Daten aus der Datenbank aus und bereitet sie für das SIAN-Modul vor.

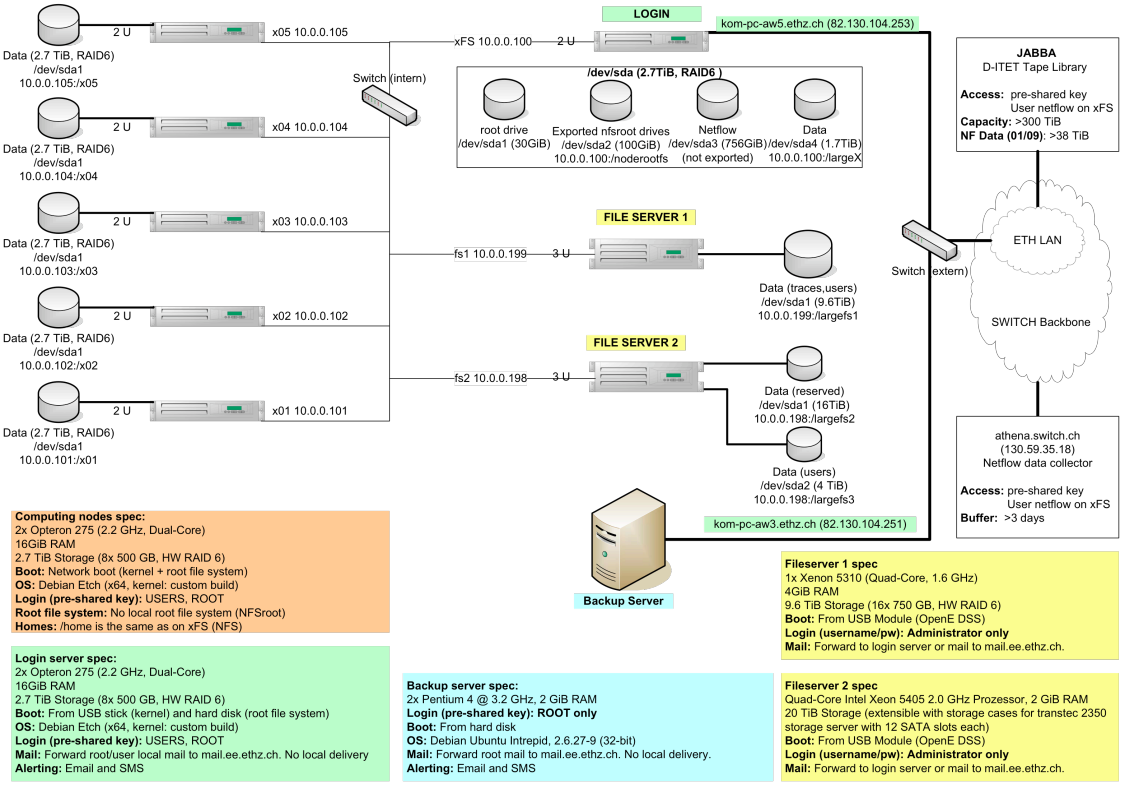
10.1.7.5. Kommunikationsschnittstelle

Die Kommunikation zwischen dem Benutzer und dem Framework, das sich auf dem Cluster befindet, wird mittels SSH auf Kommandozeilen-Ebene bewältigt.

10.1.8. Verwendete Standards (Applicable Standards)

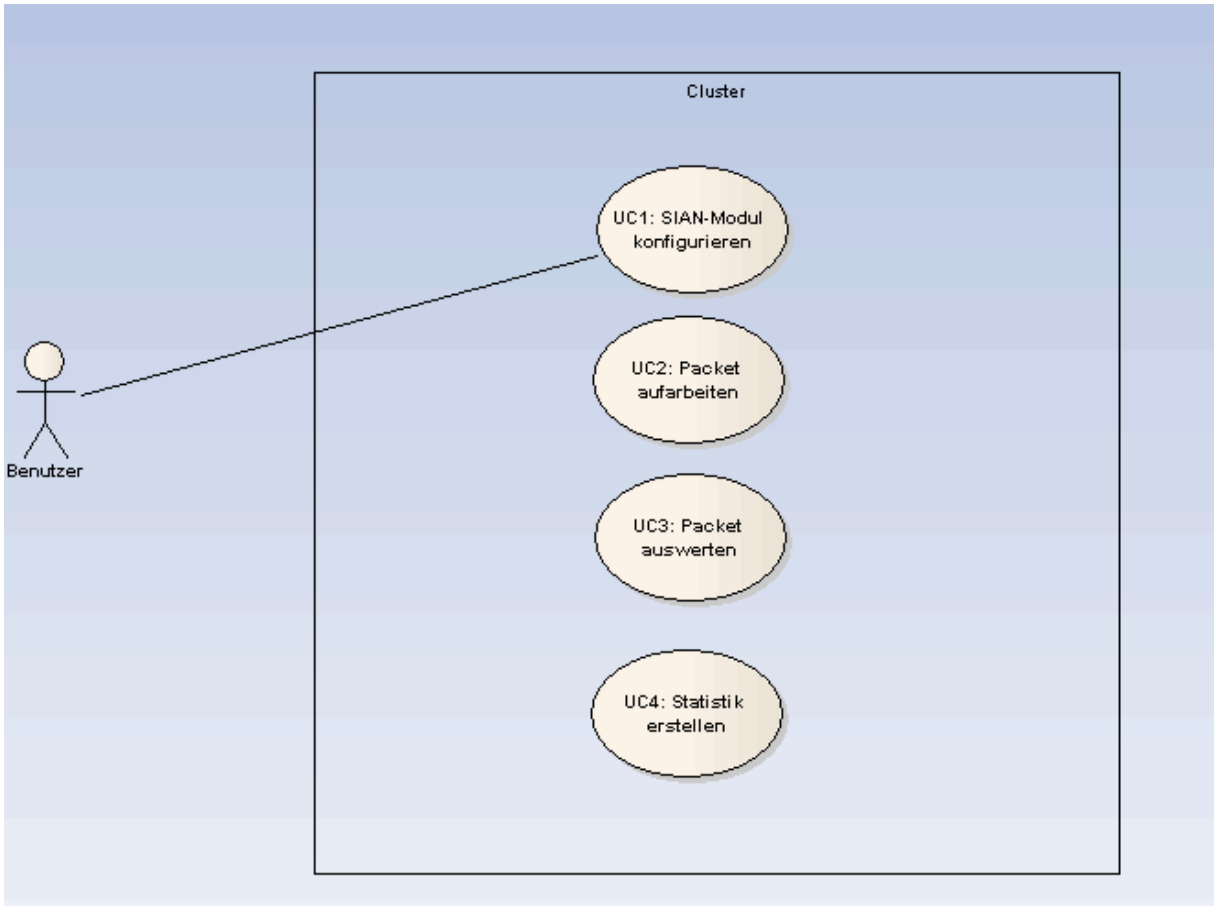
Arbeitsumgebung:	Eclipse C++
Datenbank:	JABBA-DB
Cluster:	Linux Cluster
Dokumentation:	RUP-Standards

10.1.9. Aufbau des Clusters



10.1.10. Use Cases

10.1.10.1. Use Case Diagramm



10.1.10.2. Aktoren & Stakeholders

Aktor	Beschreibung
Benutzer	Startet das Framework und konfiguriert dieses entsprechend seinen Wünschen.

Tabelle 10-1

10.1.10.3. Use Case 1: SIAN-Modul konfigurieren

UC 1	SIAN-Modul konfigurieren
Auslösender Aktor	Benutzer
Auslösende Aktion	Starten des Framework
Weitere Aktoren	System
10.1.10.4. ***Zweck****/Ziel*	Starten des Moduls mit den gewünschten Einstellungen.
Allgemeine Beschreibung	Der Benutzer möchte eine Auswertung von bestimmten Datenpaketen. Das SIAN-Modul kann entsprechend konfiguriert werden.
Anforderungen	-Framework gestartet -Cluster einsatzbereit -Daten in der Datenbank vorhanden
Fehlerfall	-Cluster nicht einsatzbereit: - Meldung ausgeben -Datenbank nicht vorhanden: -Meldung ausgeben
Kategorie	Primary
Priorität	High
Status	Essential
Detaillierungsgrad	Expanded (=„fully dressed“ gemäss Larman)
Bemerkungen	Benutzer kann das SIAN-Modul so konfigurieren, dass der Output bzw. die Statistik nach seinen Einstellungen erstellt wird.
Funktionsreferenzen	-
Offene Fragen	Keine

Tabelle 10-2

10.1.10.5. Use Case 2: Paket aufarbeiten

UC 2	Packet aufarbeiten
Auslösender Aktor	Framework
Auslösende Aktion	Starten des Framework
Weitere Aktoren	System
Zweck/Ziel	Die NetFlow-Pakete werden für die weiteren Module des Frameworks aus der Datenbank ausgelesen und entsprechend vorbereitet.
Allgemeine Beschreibung	Die Datenpakete der Datenbank werden ausgelesen und für die weiteren Module des Frameworks entsprechend angepasst und vorbereitet.
Anforderungen	-Framework gestartet -Cluster einsatzbereit -Daten in der Datenbank vorhanden
Fehlerfall	-Cluster nicht einsatzbereit: - Meldung ausgeben -Datenbank nicht vorhanden: -Meldung ausgeben -Schnittstelle zum Modul antwortet nicht: -Meldung ausgeben
Kategorie	Primary
Priorität	High
Status	Essential
Detaillierungsgrad	Expanded (=„fully dressed“ gemäss Larman)
Bemerkungen	Dieses Modul ist schon vorhanden und kann vom SIAN-Modul genutzt werden.
Funktionsreferenzen	-
Offene Fragen	Keine

Tabelle 10-3

10.1.10.6. Use Case 3: Packet auswerten

UC 3	Packet auswerten
Auslösender Aktor	Framework
Auslösende Aktion	Starten des Framework
Weitere Aktoren	System
Zweck/Ziel	Auswertung der einzelnen NetFlow-Pakete.
Allgemeine Beschreibung	Die einzelnen NetFlow-Pakete werden analysiert und kategorisiert (Web-Traffic, P2P-Traffic, Mail-Traffic, etc.).
Anforderungen	-Framework gestartet -Cluster einsatzbereit -Daten in der Datenbank vorhanden -Daten von vorgängigen Modul vorbereitet
Fehlerfall	-Cluster nicht einsatzbereit: - Meldung ausgeben -Datenbank nicht vorhanden: -Meldung ausgeben -Daten wurden nicht vorbereitet: -Meldung ausgeben
Kategorie	Primary
Priorität	High
Status	Essential
Detaillierungsgrad	Expanded (=„fully dressed“ gemäss Larman)
Bemerkungen	keine
Funktionsreferenzen	-
Offene Fragen	Keine

Tabelle 10-4

10.1.10.7. Use Case 4: Statistik erstellen

UC 4	Statistik erstellen
Auslösender Aktor	Benutzer
Auslösende Aktion	UC3
Weitere Aktoren	System
Zweck/Ziel	Ausgabe einer Statistik
Allgemeine Beschreibung	Nachdem alle NetFlow-Pakete ausgewertet wurden, wird aus den Ergebnissen eine Statistik erstellt und ausgegeben.
Anforderungen	-Framework gestartet -Cluster einsatzbereit -Daten in der Datenbank vorhanden -UC3 erfolgreich durchgeführt
Fehlerfall	-Cluster nicht einsatzbereit: - Meldung ausgeben -Datenbank nicht vorhanden: -Meldung ausgeben
Kategorie	Primary
Priorität	High
Status	Essential
Detaillierungsgrad	Expanded (=„fully dressed“ gemäss Larman)
Bemerkungen	Benutzer kann das SIAN-Modul so konfigurieren, dass der Output bzw. die Statistik nach seinen Einstellungen erstellt wird.
Funktionsreferenzen	-
Offene Fragen	Keine

Tabelle 10-5

10.2. Domainanalyse

10.2.1. Einführung (Introduction)

10.2.1.1. Zweck

Dieses Dokument legt das Design der zu implementierenden Applikation fest. Zweck des Dokuments ist die Definition der entscheidenden Lösungsvarianten.

10.2.1.2. Gültigkeitsbereich

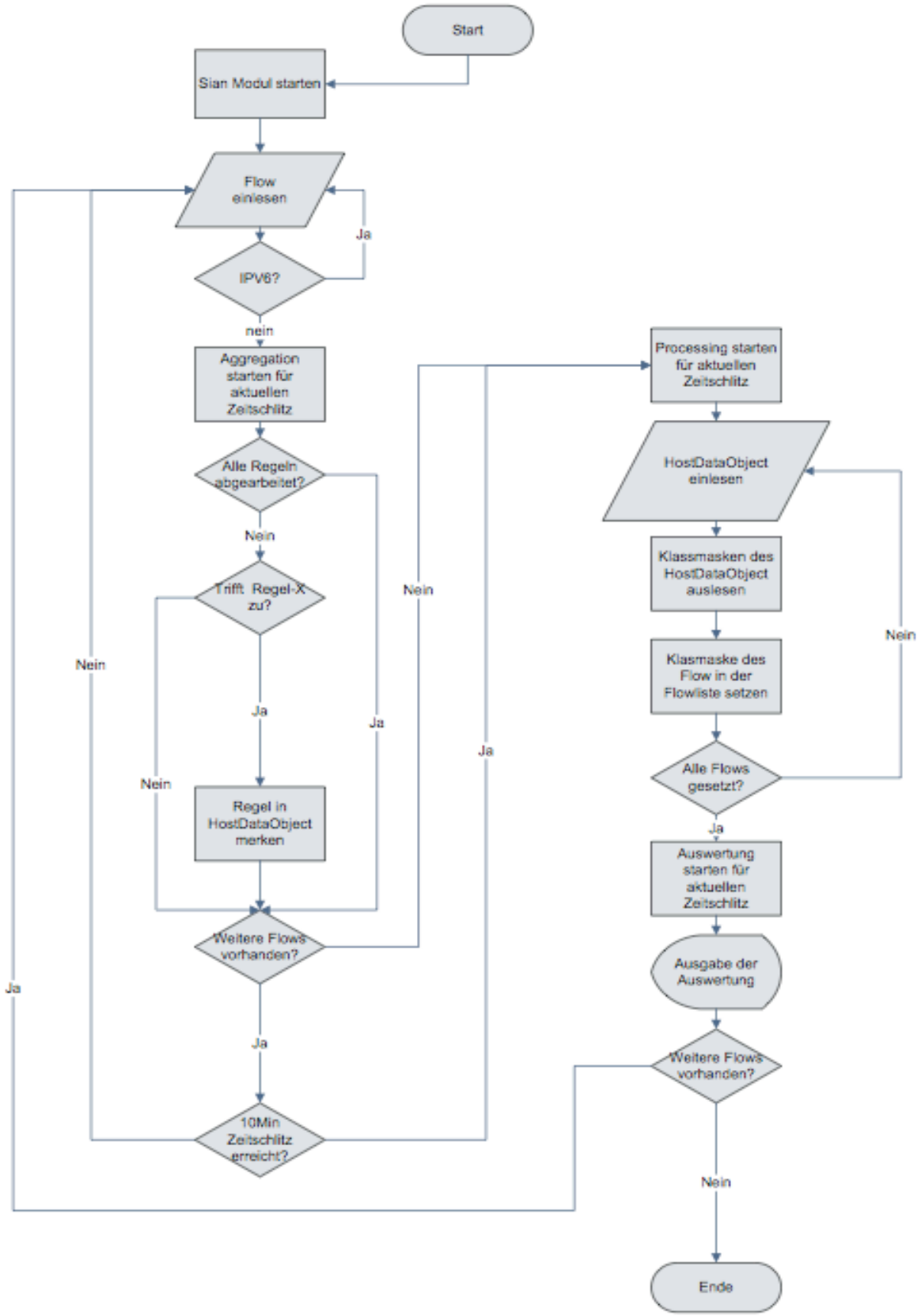
Dieses Dokument bestimmt das Projekt SIAN als Semesterarbeit im 6. Semester der Abteilung Informatik. Das Modul ist Teil eines Frameworks, das zur Analyse des schweizerischen Internetverkehrs dient.

10.2.1.3. Definitionen und Abkürzungen

Siehe Fachbericht Kapitel Glossar

10.2.2. Domain Modell

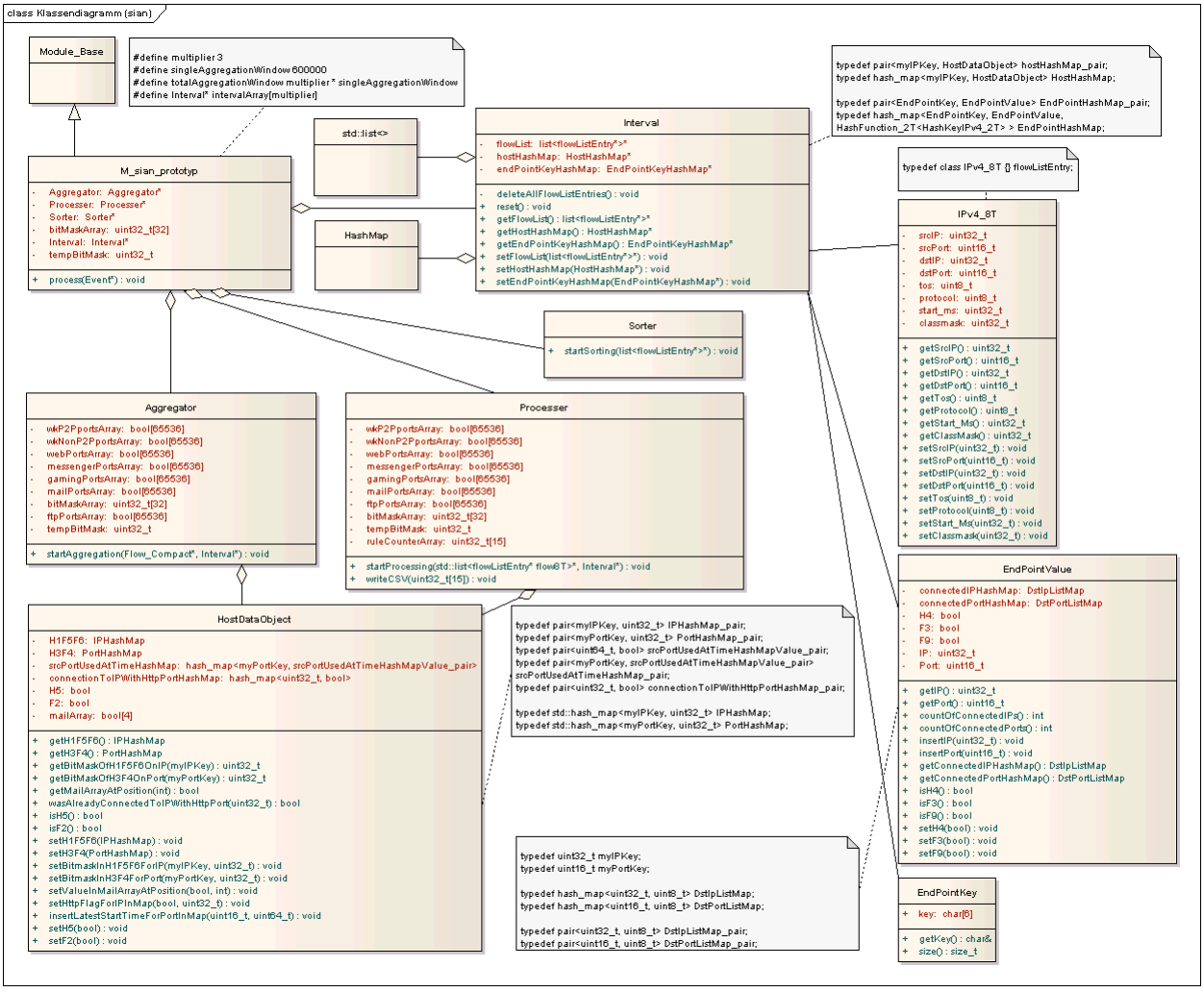
10.2.2.1. Flussdiagramm



10.2.3. Klassen

Das folgende Klassendiagramm beschreibt die genauere Klassenarchitektur unseres Projektes.

10.2.3.1. Klassendiagramm



10.2.3.2. Beschreibung

1	**M_sian_prototyp**	
Beschreibung	Hauptklasse erbt Eigenschaften von Module_Base.	
Attribute	Aggregator : Aggregator*	Datenaggregator.
	Processor : Processor*	Datenevaluierer.
	Sorter : Sorter*	Sortierfunktion
	bitMaskArray : uint32_t[32]	Array mit den Bitmasken
	Interval : Interval*	Zeiger auf das aktuelle Interval
	tempBitMask : uint32_t	Hilfsbitmaske als Hilfe um die entsprechende Bits zu setzen
Methoden	process(Event* event) : void	Wird vom vorherigen Modul mit der aktuellen Dateiliste aufgerufen.
Beziehungen	Beinhaltet 1 Aggregator-, 1 Interval-, 1 Processer-Objekt und 1 globale Flow-Liste (globalFlowList).	

2	**Interval**	
Beschreibung	Interval Klasse mit all Informationen die pro Interval benötigt werden.	
Attribute	flowList ; list<flowListEntry*>*	List mit den Flows.
	hostHashMap HostHashMap*	Hashmap mit den Hosts pro Interval
	endPointKeyHashMap : EndpointKeyHaschMap*	Pointer auf einen EndPoint
Methoden	deleteAllFlowListEntries() : void	Löscht am Ende eines Intervall alle Flows
	reset() : void	Setzt alles wieder zurück
	getFlowList() : list<flowListEntry*>*	Gibt eine List mit den Flows zurück
	getEndPointKeyHashMap : EndPointKeyHash*	Gibt die EndPoint bezogene HashMap zurück
	setFlowList(list<flowListEntry*>*) : void	Setzt pro Interval eine FlowListe
	setHostHashMap(HostHashMap*) : void	Setzt pro Interval eine FlowHashMap
	setEndPointKeyHashMap(End PointKeyHashMap*) : void	Setzt pro Interval eine EndPoint bezogene Hashmap.
Beziehungen	Wird in den Klassen M_sian_prototyp, IPv4_8T, EndPointValue und EndPointKey verwendet.	

3	IPv4_8T	
Beschreibung	8 Tupel-Klasse mit den wichtigsten Flow-Informationen.	
Attribute	srcIP : uint32_t*	Source-IP.
	srcPort : uint16_t*	Source-Port.
	dstIP : uint32_t*	Destination-IP.
	dstPort : uint16_t*	Destination-Port.
	tos : uint8_t*	Type of Service.
	protocol : uint8_t*	Transport-Protokoll (TCP/ UDP).
	start_ms : uint32_t*	Start-Zeit in Millisekunden.
	classmask : uint32_t*	Klassenmaske (Bitmaske).
Methoden	Getter-Methoden: uintX_t*	Getter-Methoden für den Lese-Zugriff auf die einzelnen Attribute.
	Setter-Methoden: void	Setter-Methoden für den Schreib-Zugriff auf die einzelnen Attribute.
Beziehungen	Wird in der Klasse M_sian_prototyp verwendet, die Einträge in der globalen Flow-Liste zu definieren.	

4	Aggregator	
Beschreibung	Aggregator-Klasse ist zuständig für die komplette Datenaggregierung.	
Attribute	wkP2PportsArray : bool[65536]	Bekannte P2P-Ports.
	wkNonP2PportsArray : bool[65536]	Bekannte NonP2P-Ports.
	webPortsArray : bool[65536]	Bekannte Web-Ports.
	messengerPortsArray : bool[65536]	Bekannte Messenger-Ports.
	gamingPortsArray : bool[65536]	Bekannte Gaming-Ports.
	mailPortsArray : bool[65536]	Bekannte Mail-Ports.
	bitMaskArray : unint32_t[32]	BitMasken Arraky um die einzelnen Bitmask pro Host zu speichern.
	tempBitMask : uint32_t	Hilfsbitmaske um die einzelnen Bitmasken pro Host zu setzen.
	ftpPortsArray : bool[65536]	Bekannte FTP-Ports.
Methoden	startAggregation(Flow_Compact*, Interval*)	Aggregierungs-Methode, die das originale, direkt übergebene HostHashMap-Objekt verändert bzw. aggregiert.
Beziehungen	Beinhaltet eine Assoziation zur HostDataObject-Klasse.	

5	**Processer**	
Beschreibung	Processer-Klasse ist zuständig für die komplette Datenevaluierung.	
Attribute	wkP2PportsArray : bool[65536]	Bekannte P2P-Ports.
	wkNonP2PportsArray : bool[65536]	Bekannte NonP2P-Ports.
	webPortsArray : bool[65536]	Bekannte Web-Ports.
	messengerPortsArray : bool[65536]	Bekannte Messenger-Ports.
	gamingPortsArray : bool[65536]	Bekannte Gaming-Ports.
	mailPortsArray : bool[65536]	Bekannte Mail-Ports.
	bitMaskArray : unint32_t[32]	BitMasken Arraky um die einzelnen Bitmask pro Host zu speichern.
	tempBitMask : uint32_t	Hilfsbitmaske um die einzelnen Bitmasken pro Host zu setzen.
	ftpPortsArray : bool[65536]	Bekannte FTP-Ports.
Methoden	startProcessing(std::list<flowListEntry&>&, Interval*) : void	Processing-Methode, die das originale, direkt übergebene HostHashMap-Objekt liest und die entsprechenden Bitmasken in der originalen, direkt übergebenen globalen Flow-Liste setzt.
	csvWrite(unint32_t[15]) : void	Methode, die zuständig ist für den CSV-Export der Statistik-Daten.
Beziehungen	Beinhaltet eine Assoziation zur HostDataObject-Klasse.	

6	**HostDataObject**	
Beschreibung	HostDataObject beinhaltet die aggregierten, hostspezifischen Informationen, nach jedem 10 Minuten - Intervall bereit um vom Processer evaluiert zu werden.	
Attribute	H1F5F6 : IPHashMap*	HashMap, welche die für die Regeln H1, F5 und F6 notwendigen Informationen beinhaltet.
	H3F4 : PortHashMap*	HashMap, welche die für die Regeln H2, H3 und F4 notwendigen Informationen beinhaltet.
	srcPortUsedAtTimeHashMap : hash_map<myPortKey, srcPortUsedAtTimeHashMapValue _pair>	Hashmap, das die scrPort mit der letzten darauf zugegriffenen Zeit speichert.
	connectionToIPWithHttpPortHashMap : hash_map<uint32, bool>	HashMap, das die Destination-IP und speichert und sich merkt, ob schon einmal zu dieser IP über ein Http-Port kommuniziert wurde.
	H5 : bool	Flag für die H5 Regel
	F2 : bool	Flag für die F2 Regel
	mailArray : bool[4]	Bool-Array für die Mail-Regel (smtp_send, smtp_reicv, pop, imap).
Methoden	Getter-Methoden	Getter-Methoden für den Lese-Zugriff auf die einzelnen Attribute.
	Setter-Methoden	Setter-Methoden für den Schreib-Zugriff auf die einzelnen Attribute.
	wasAlreadyConnectedToIPWithHttpPort(uint32_t) : bool	Gibt zurueck ob Host schon einmal mit der Destination-IP über ein Http-Port kommuniziert hat.
	isH5() : bool	Ob Host H5 ist.
	isF2() : bool	Ob Host F2 ist.
	insertLatestStartTimeForPortInMap(uint16_t portParam, uint64_t startTimeParam) : void	Fügt späteste Startzeit ein.
Beziehungen	Beinhaltet eine Assoziation zur HashKeyIPv4_2T-Klasse.	

7	**EndPointValue**	
Beschreibung	Beinhaltet EndPoint bezogene Werte	
Attribute	connectedIPHashMap : DstIPListMap	Liste mit den IP auf denen eine Verbindung aufgebaut wurde.
	connectedPortHashMap : DstPortListMap	Liste mit den Port auf denen eine Verbindung aufgebaut wurde.
	H4 : bool	Flag für die H4 Regel.
	F3 : bool	Flag für die F3 Regel.
	F9 : bool	Flag für die F9 Regel.
	IP : unint32_t	Speichert die IP eines Flows.
	Port : uint16_t	Speichert den Port eines Flows.
Methoden	getIP() : uint32_t	Gibt die IP eines EndPoints zurück
	getPort() : uint16_t	Gibt den Port eines EndPoints zurück
	countOfConnectedIPs() : int	Anzahl vom EndPoint aus verbundenen IPs.
	countOfConnectedPors() : int	Anzahl vom EndPoint aus verbundenden Ports.
	insertIP(uint32_t) : void	Setzt die IP eines EndPoints.
	insertPort(uint16_t) : void	Setzt den Port eines EndPoints.
	getConnectedIPHashMap() : DstIPListMap	Gibt die Liste mit dem vom EndPoint aus verbundenen IPs.
	getConnectedPortHashMap() : DstPortListMap	Gibt die Liste mit dem vom EndPoint aus verbundenen Ports.
	isH4() : bool	Gibt TRUE zurück falls die H4 Regel zugeschlagen hat.
	isF3() : bool	Gibt TRUE zurück falls die F3 Regel zugeschlagen hat.
	isF9() : bool	Gibt TRUE zurück falls die F9 Regel zugeschlagen hat.
	setMethoden	Setzt die Attribute.
Beziehungen	Beinhaltet eine Beziehung zur Klasse Interval	

8	**EndPointKey**	
Beschreibung	Stellt den Schlüssel eines EndPoint dar	
Attribute	Key : char[6]	Ist der Schlüssel eines EndPoints.
Methoden	getKey() : char&	Gibt die Referenz auf den Schlüssel.
	Size(): size_t	Gibt die Grösse eines EndPoints zurück
Beziehungen	Beinhaltet eine Beziehung zur Klasse Interval	

9	**Sorter**	
Beschreibung	Klasse zur Sortierung der Flows nach der Startzeit.	
Methoden	startSorting(list<flowListEntry*>*) :void	Sortiert die Flows gemäss ihrer Startzeit aufsteigend.
Beziehungen	Beinhaltet eine Beziehung zur Klasse M_sian_prototyp	

10.3. Zeit- und Raumkomplexitäts Analyse

10.3.1. Einführung (Introduction)

10.3.1.1. Zweck

Dieses Dokument zeigt die Raum- und Zeitkomplexität für das Projekt SIAN. Es werden die verschiedenen Zeiten, wie Zugriff, Ausgabe, etc. berechnet und es wird aufgezeigt was für ein Speicher- und Zeitverbrauch beansprucht wird.

10.3.1.2. Gültigkeitsbereich

Dieses Dokument bestimmt das Projekt SIAN als Semesterarbeit im 6. Semester der Abteilung Informatik. Das Modul ist Teil eines Frameworks, das zur Analyse des schweizerischen Internetverkehrs dient.

10.3.1.3. Definitionen und Abkürzungen

Siehe Fachbericht Kapitel Glossar.

10.3.2. Zeitkomplexität

10.3.2.1. Einleitung

Für die Bestimmung der Zeitkomplexität haben wir uns auf folgende Referenzen gestützt:

-Buch: Data Structures & Algorithms in Java, 4th Edition

-Testdaten: testdata_v5/19993_20V5flowsReference_0.dat.bz2

-Aussage von Herrn Tellenbach: 10min = 20 Mio Flows

10.3.2.2. Daten und Fakten

10.3.2.2.1. Zugriffzeiten HashMap & List

Methode	List	HashMap
size, isEmpty	O(1)	O(1)
entries	0(n)	O(n)
find	0(n)	O(1) und O(n) worst-case
findAll	0(n)	O(1 + s) und O(n) worst-case
insert	0(1)	O(1)
remove	0(n)	O(1) und O(n) worst-case

10.3.2.2.2. Testdaten

Von den obengenanten Testdaten konnten folgende Informationen ausgelesen werden:

-Anzahl Flows: 10

-Start-Zeit 1. Flow: 1’193’288’363’950 ms

-End-Zeit letzter Flow: 1'193'289'808’814 ms

-Total Dauer: 1'444'864 ms → 1'444,864 sek → 24,08 min

10.3.3. Berechnung

10.3.3.1. Informationen

-In 120Mio. Flows hat es im Durschnitt ca. 7 Mio. Hosts.

-In 20Mio Flows hat es im Worst-Case 20Mio-EndPoints und 40Mio. IP’s.

-Anzahl EndPoints = 1/10 der Flows.

10.3.3.2. Berechnung:

Anzahl Flows in 10min:	20'000’000 Flows
Anzahl EndPoints in 10min:	2'000'000 EndPoints
Anzahl Zugriffe auf 1 Flow:	77
Anzahl Zugriffe auf 1 EndPoint pro Flow:	2
Anzahl Zugriffe auf HashMap (find):	46
Anzahl Zugriffe auf HashMap (add):	18

Total Dauer = 20'000’000 * (77xO(n) + 2’000’000xO(1) + 46xO(1) + 18xO(1)) Worst-Case

20'000’000 * (77xO(1) + 2’000’000xO(1) + 46xO(1) + 18xO(1)) Best-Case

10.3.4. Raumkomplexität

10.3.4.1. Einleitung

Für die Bestimmung der Raumkomplexität haben wir uns auf folgende Referenzen gestützt:

-Buch: Data Structures & Algorithms in Java, 4th Edition

-Testdaten: testdata_v5/19993_20V5flowsReference_0.dat.bz2

10.3.4.2. Daten und Fakten

10.3.4.2.1. Speicherverbrauch HashMap & List

Grösse	List	HashMap
n Byte	n Byte	n Byte + Overhead (1,5Kb)

10.3.4.2.2. Testdaten

Von den obengenanten Testdaten konnten folgende Informationen ausgelesen werden:

-Totale Anzahl Flows: 10

-Grösse eines Flows: 86 Byte

-durchschnittliche Grösse 1 HostDataObject: 94 Byte

-Anzahl Hashmaps: 6144 Byte (4x1536)

10.3.4.3. Berechnung:

Anzahl Flows in 10min: 20'000’000 Flows

Total Speicher = 20'000’000 * 86 + 20'000’000* 94 = 3’600’000’000 Byte

+ 6144Bypte = 3'600'006'144 Byte

= 3'515’631 KB

= 3’433 MB

10.4. Projektplan

10.4.1. Einführung

10.4.1.1. Zweck

Dieses Dokument beinhaltet die Übersicht über die Planung und Organisation des Projektes. Es werden die Management Abläufe und Risiken, Aufteilung der Arbeitspakete, sowie die Qualitätsmassnahmen beschrieben.

10.4.1.2. Gültigkeitsbereich

Dieses Dokument ist für die gesamte Projektzeit gültig. Änderungen bleiben vorbehalten.

10.4.1.3. Definitionen und Abkürzungen

Das Glossar mit den Definitionen und Abkürzungen wird während des Projektes fortlaufend erweitert.

Siehe „Glossar.pdf“

10.4.1.4. Referenzen

- Anforderungsspezifikationskatalog „Anforderungsspezifikation.pdf“
- Notizen und Unterlagen der Sitzungen im Ordner „Unterlagen“
- Literaturverzeichnis: Siehe „http://cvsi.hsr.ch/svn/sian“

10.4.2. Projekt Übersicht

An der ETH Zürich werden zu Forschungszwecken Datenanalysen an realen Verkehrsdaten eines mittelgrossen schweizerischen Internet Service Providers (ISP) durchgeführt. Dazu steht eine grosse Datensammlung (komprimiert ca. 29 TB) von CISCO NetFlow-Daten zur Verfügung, die fortlaufend um die neuesten Verkehrsinformationen ergänzt werden. Um diese Daten zu verarbeiten stehen ein leistungsfähiges Software Framework sowie ein moderner Computer Cluster zur Verfügung.

10.4.3. Zweck und Ziel

Das Ziel dieser Arbeit besteht darin, das bestehende Framework um ein Modul zu ergänzen, das die Identifikation verschiedener Applikationsklassen unterstützt. Zur Umsetzung der entsprechenden aus der Literatur bekannten Heuristiken ist die Wahl effizienter Datenstrukturen und Algorithmen eminent wichtig, da nur so die grossen Datenmengen in endlicher Zeit verarbeitet werden können. Dazu sind die Heuristiken zuerst zu analysieren, bevor die Erstellung der Software in C++ in Angriff genommen wird.

10.4.4. Annahmen und Einschränkungen

Wir gehen von folgenden Annahmen und Einschränkungen aus:

- Das Einarbeiten in das bestehende Framework dauert nicht länger als 3 Wochen.
- Die Schnittstellen zu den anderen Modulen sind verfügbar und dokumentiert.
- Die Richtzeit pro Mitglied wird auf 16 Stunden die Woche angenommen.

10.4.5. Projektorganisation

10.4.5.1. Organisationsstruktur

Projektmitglied	Verantwortung	Aufgaben
Selim Akyol	Entwickler	- Evaluation -Processierung
Edon Berisha	Entwickler	- Analyse -Aggregation

10.4.6. Externe Schnittstellen

Ansprechpartner	Bernhard Tellenbach, ETH
Verantwortlicher	Eduard Glatz
Betreuer	Eduard Glatz Bernhard Tellenbach

10.4.7. Management Abläufe

10.4.7.1. Projekt Kostenvoranschlag

Das Projekt startet Mitte Februar 2009 und die funktionsfähige Version muss Ende Mai 2009 abgeliefert werden. Falls der Arbeitsaufwand grösser wäre als geplant, werden mehr als die durchschnittlichen Stunden am Projekt gearbeitet.

10.4.8. Projektplan

10.4.8.1. Zeitplan

Siehe Kapitel „Zeitplan“

10.4.8.2. Iterationsplanung / Meilensteine

Iterationsplanung		
Von	**Bis**	**Phase**
16.02.09	15.03.09	Inception
16.03.09	29.03.09	Elaboration Iteration 1
30.03.09	12.04.09	Elaboration Iteration 2
13.04.09	26.04.09	Construction Iteration 1
27.04.09	10.05.09	Construction Iteration 2
28.04.09	12.05.09	Transition
13.05.09	25.05.09	Reserve

Meilensteine			
Nr.	**Titel**	**Beschreibung**	**Datum**
1	Projektplan	Projektplan, Zeitplan, Meilensteine fertiggestellt	29.02.09
2	Anforderung und Analyse	Anforderungsspezifikation und Domainanalyse	12.03.09
	Proof of Concept	Erste Programmierarbeit: Auslesen eines NetFlow Pakets & Map der Interfaces auf den Routern	
3	Design	Internes Design und Designdokumentation vollständig	01.04.09
4	Construction	Entwicklung am Code abgeschlossen	29.04.09
5	Transition	Tests erfolgreich abgeschlossen, Dokumentation vollständig	13.05.09
6	Schlussabgabe	Abgabe und Präsentation	29.05.09

10.4.8.3. Meetings

Regelmässige Besprechungen finden in der Regel jede Woche jeweils am Donnerstag statt und sonst gemäss Absprache mit den Dozenten.

10.4.8.4. Releases

Realease	**Datum**	**Funktionalität**
Prototyp	01.04.09	Grobe Funktionalität
Alpha	11.04.09	Hauptalgorithmus implementiert
Beta	22.05.09	Code vollständig Implementiert mit Codeunschönheiten,
Final	29.05.09	Codeunschönheiten beseitigt, letzte Anpassungen vorgenommen, Modul einsatzbereit

10.4.9. Risiko Management

10.4.9.1. Skalendefinition

Auswirkung (AW)

1-10 (1= schwach, 5 = mittel, 10 = stark)

Eintrittswahrscheinlichkeit (EW)

1%-100%

Wichtigkeit (WI)

0.1-1.0 (0.1 = unwichtig, 1.0 = sehr wichtig)

10.4.10. Risikoanalyse

Risiko	AW	WI	EW	Massnahmen
Projekt				
Zeitplan kann nicht eingehalten werden	7	0.8	40%	Arbeitspaket aus Scope entfernen/ Genug Reservezeit.
Modul kann Schnittstelle nicht benutzen	9	0.8	15%	Schnittstelle vor Modul-Implementierung genau analysieren.
Evaluierungsalgorithmus braucht zu viel Zeit	5	0.4	60%	Genaue Messung und Berechnung von verschiedenen Algorithmen um den besten herauszufinden.
Projektbeteiligte				
Paket-Auswertung zu komplex	6	0.8	26%	Einschulung mit Bücher, Internet und anderen Ressourcen und durch Hilfeleistung der Betreuer.
Mitglied kann sein Arbeitspaket nicht erledigen	4	0.6	44%	Neuzuweisung des Arbeitspaketes/ Betroffener Beteiligter erledigt anderes Arbeitspaket.
Mitglied steigt aus Projekt aus	10	0.8	3%	Umstrukturierung der Verantwortlichkeiten für die Arbeitspakete des Ausgestiegenen.
Projektumfeld				
Komponenten des Testumfeldes funktionieren nicht	8	0.7	20%	Genaue Analyse der Komponenten und Hilfeleistung durch Betreuer.
Änderungswünsche der Betreuer	4	0.6	65%	Wenn Änderung nicht zu gross, dann Änderung vornehmen, sonst ablehnen.

10.4.11. Arbeitspakete

Weitere Arbeitspakete werden am Ende jeder Iteration in den Teammeetings definiert, geschätzt und gemäss den Milestones zugeteilt.

1. Inception

1.1	Projektplan	alle	10 h
Beschreibung	Projektplan gemäss Vorlage erstellen und einreichen. Projektorganisation festlegen und Iterationen / Meilensteine planen.		
Abhängigkeiten			
Risiken	-		
Artefakte	Projektplan.doc		

1.2	Projektplan erweitern	alle	5 h
Beschreibung	Projektplan erweitern, Arbeitspakete weiter unterteilen, Verantwortlichkeiten zuteilen.		
Abhängigkeiten	[1.2] Projektplan erstellt.		
Risiken	-		
Artefakte	Projektplan.doc		

1.3	Anforderungsanalyse	seak/edbe	16 h
Beschreibung	Anforderungsanalyse mit Betreuer durchführen und in Pflichtenheft festhalten.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	Anforderungsspezifikation.doc		

1.4	Define Use Cases	seak/edbe	2 h
Beschreibung	Use Cases definieren.		
Abhängigkeiten	Anforderungsanalyse [1.4]		
Risiken	-		
Artefakte	Anforderungsspezifikation.doc		

1.5	Aufbau Testumgebung	seak/edbe	8 h
Beschreibung	Installation aller benötigten Soft- und Hardware.		
Abhängigkeiten	Anforderungsanalyse [1.4]		
Risiken	-		
Artefakte	-		

1.6	Proof of Concept	seak/edbe	9 h
Beschreibung	Testprogramm schreiben: Auslesen eines NetFlow Pakets & Map der Interfaces		
Abhängigkeiten	Aufbau Testumgebung [1.6]		
Risiken	-		

Artefakte	C++-Projekt auf dem SVN-Server

1.8	**Glossary**	**edbe**	**1 h**
Beschreibung	Glossary beginnen.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	Glossar.doc		

2. Elaboration

2.1	**Speicherverfahren (HashMaps, 1. Diskussionsstoff)**	**seak & edbe**	**10 h**
Beschreibung	Modellierung der ersten HashMaps zur Speicherung der NetFlow-Daten.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	PD & Klassendiagramm in UML, Domainanalyse.doc		

2.2	**Analyse des Frameworks (Klassenhierarchie, Kernmethoden)**	**seak & edbe**	**30 h**
Beschreibung	Modellierung der wichtigsten Klassen und Methoden.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	PD & Klassendiagramm in UML, Domainanalyse.doc		

2.3	**Domain Analyse**	**seak & edbe**	**9 h**
Beschreibung	PD und Klassendiagramm in UML.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	PD & Klassendiagramm in UML, Domainanalyse.doc		

2.3.1	**Problem Domain**	**seak & edbe**	**9 h**
Beschreibung	PD: Mögliche Klassen und verantwortliche Methodenimplementierungen analysieren. Schnittstellen (Methoden, Formate, etc.) analysieren und für weitere Designentscheidungen berücksichtigen. Analyse der PD.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	PD & Klassendiagramm in UML, Domainanalyse.doc		

2.4	**Speicherstrukturen & Design**	**seak & edbe**	**27 h**
Beschreibung	Implementierungsentscheide & Klassendiagramm in UML.		

Abhängigkeiten	-
Risiken	-
Artefakte	PD & Klassendiagramm in UML, Domainanalyse.doc

2.4.1	**Implementierungsentscheidung & -modellierung (Speicher- und Zeitverbrauch)**	**seak & edbe**	**24 h**
Beschreibung	**Analyse der Implementierungsmöglichkeiten der Heuristiken (HashMap, HashTable, (Priority-) Queue, Array, ArrayList, Vector, etc.). Schreibe- und Ausleseverfahren.**		
Abhängigkeiten	**-**		
Artefakte	PD & Klassendiagramm in UML, Domainanalyse.doc		

2.4.2	**Klassendiagramm**	**edbe**	**3 h**
Beschreibung	**Klassendiagramm in UML designen.**		
Abhängigkeiten	**-**		
Risiken	**-**		
Artefakte	Klassendiagramm in UML		

2.5	**Refinement of Use Cases**	**seak & edbe**	**4 h**
Beschreibung	UCs genau beschreiben.		
Abhängigkeiten	Define Use Caes [1.4]		
Risiken	-		
Artefakte	Anforderungsspezifikation.doc & Domainanalyse.doc		

2.5.1	**All Ucs Fully dressed**	**seak**	**2 h**
Beschreibung	Alle UCs im fully dessedformat beschreiben.		
Abhängigkeiten	Define Use Caes [1.4]		
Risiken	-		
Artefakte	Anforderungsspezifikation.doc		

2.5.2	**SSD & Contracts**	**edbe**	**2 h**
Beschreibung	SSD & Contracts zu allen UCs.		
Abhängigkeiten	Define Use Caes [1.4]		
Risiken	-		
Artefakte	Domainanalyse.doc		

2.6	Coding (1. Prototyp)	edbe & seak	64 h
Beschreibung	Ersten Prototypen mit Speicherung und Auswertung.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	Erster Prototyp (C++-Projekt auf SVN-Server)		

2.6.1	Problem Domain	edbe & seak	64 h
Beschreibung	Wichtigste Klassen und mindestens eine Heuristikabdeckung (z.B. P2P, Attacken, etc. inkl. CSV-Export oder Konsolenoutput) implementieren.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	Erster Prototyp (C++-Projekt auf SVN-Server)		

2.7	Testen		6 h
Beschreibung	Schreiben von Modultests und Testen der Hauptfunktionalität.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	Testprotokoll.doc		

2.8	Demonstration und Feedback		2 h
Beschreibung	Demonstration des Prototypen bei den Betreuer und Feedback einholen.		
Abhängigkeiten	-		
Risiken	-		
Artefakte	Feedback1.doc		

3. Construction Iteration 1

3.1	Domain Model	Seak Edbe	5
Beschreibung	PD und Klassendiagramm anpassen gemäss Betreuerfeedback.		
Abhängigkeiten	Demonstration und Feedback [2.6]		
Risiken	-		

3.2	Coding	Seak Edbe	50
Beschreibung	Entwicklung der Anpassungen.		
Abhängigkeiten	-		
Risiken	-		

3.2.1	Problem Domain	Seak Edbe	50
Beschreibung	Umsetzen der PD-Anpassungen.		
Abhängigkeiten	Domain Model [3.1]		
Risiken	-		

3.3	Testen	Seak Edbe	0
Beschreibung	Prototyptest.		
Abhängigkeiten	-		
Risiken	-		

3.4	Demonstration und Feedback	Seak Edbe	10
Beschreibung	Demonstration des Prototypen bei den Betreuer und Feedback einholen.		
Abhängigkeiten	-		
Risiken	-		

4. Construction Iteration 2

4.1	Domain Model		4
Beschreibung	PD und Klassendiagramm anpassen gemäss Betreuerfeedback.		
Abhängigkeiten	Demonstration und Feedback [3.4]		
Risiken	-		

4.1	Coding		30
Beschreibung	Entwicklung des kompletten Moduls.		
Abhängigkeiten	-		
Risiken	-		

4.1.1	Problem Domain		30
Beschreibung	Umsetzen der PD-Anpassungen.		
Abhängigkeiten	Domain Model [4.1]		
Risiken	-		

4.2	Demonstration und Feedback		10
Beschreibung	Demonstration des Prototypen bei den Betreuer und Feedback einholen.		
Abhängigkeiten	-		
Risiken	-		

5. Transition

5.1	Integration des Moduls in das Framework und den scharfen Daten	unkown	0
Beschreibung	Integration des Moduls in das Framework, testen und Abnahme.		
Abhängigkeiten	-		
Risiken	-		

5.2	Code Review	Seak Edbe	25
Beschreibung	Code Review und nächsten Release planen. (nach dem Projekt)		
Abhängigkeiten	-		
Risiken	-		

5.3	Abgabe	Seak Edbe	40
Beschreibung	Alle nötigen Artefakte für die Abgabe vorbereiten, zusammenstellen und abgeben.		
Abhängigkeiten	-		
Risiken	-		

10.4.12. Infrastruktur

10.4.12.1. Koordination

Zur Synchronisation der Projektentwicklung verwenden wir einen SVN-Server. Dieser dient zugleich für die Datenablage des Java-Projektes sowie als Ablage für die Projektdokumentation.

10.4.13. Arbeitsumgebung

Folgende Hard- & Software brauchen wir während der Entwicklung unseres Projektes in unserer Testumgebung.

10.4.13.1. Hardware

Gerät	Hersteller	Schnittstelle
Laptops	APPLE	-
Cluster der ETH	-	SSH

10.4.13.2. Software

	Name	Hersteller	Version
Betriebssysteme	Windows, MAC OS	Microsoft/ Apple	XP/ X, Leopard
	Windows Server	Microsoft	2003
Entwicklungsumgebung	Eclipse	-	3.2.2
	Subclipse	-	1.2
Office	Word	Microsoft	200x
	Excel	Microsoft	200x

10.4.14. Qualitätsmassnahmen

Um für unser Produkt sowie dem gesamten Projektverlauf eine möglichst hohe Qualität zu erreichen, werden folgende Massnahmen unternommen.

10.4.15. Richtlinien

Die verwendeten Coderichtlinien basieren auf den Standarteinstellungen des Code Formatters von Eclipse.

10.4.16. Code

- Alle Namen, wie Variablen und Methoden sind in Englisch.
- Kommentare sind in Deutsch gefasst.
- Tests werden mit CUTE durchgeführt.

10.4.17. Codereviews

- Wöchentlich bzw. bei jedem Teammeeting wird ein Codereview vom Team durchgeführt.
- Testprotokolle werden in den Teammeetings besprochen.

10.4.18. Usability-Test

- Diese werden durch den Betreuern durchgeführt und von uns bearbeitet.

10.4.19. Prozessqualität

10.4.19.1. Diskussionen und Reviews

Die Projektmitglieder und Betreuer kommen einmal in der Woche zusammen um über erreichte Ziele und den nächsten nötigen Schritten zu diskutieren. Dabei schätzen wir den benötigten Aufwand für die nächsten Schritte ein und schlagen Realisierungs- und Vorgehensmöglichkeiten vor. Erreichte Ziele und unternommene Schritte werden kurz diskutiert. Wo nötig werden Verbesserungsvorschläge eingebracht und neue Termine festgelegt.

Entscheidungen werden schriftlich in Form von Sitzungsberichten festgehalten und, wo nötig, in die entsprechenden Stellen der Dokumentation aufgenommen. Die Sitzungsberichte werden nur für interne Zwecke verwendet, weshalb sie nicht zur Dokumentation gehören.

Um Missverständnisse unter den Projektmitgliedern zu vermeiden, finden unter der Woche, je nach bedarf, Kurzdiskussionen von ca. 5-10 min statt.

10.4.20. Management und Verwaltung

Arbeitspakete und Teildokumentationen werden so verteilt, dass sich die Teammitglieder nicht überschneiden und/oder gegenseitig blockieren können. Sollte es dennoch zu Überschneidungen oder Blockaden kommen, wird die Arbeitsaufteilung neu diskutiert und aufgeteilt.

10.4.21. Dokumentation

Die Dokumentation wird je nach Aufgabenzuteilung aufgeteilt. Jeder erstellt die Teildokumentation für sich. Andere Teammitglieder können Verbesserungsvorschläge bringen oder auf Fehler hinweisen. Jedoch ist der Autor für die Überarbeitung zuständig. Zum Schluss werden die einzelnen Teile in ein Dokument von einem Mitglied zusammengenommen und von den übrigen gelesen. Überarbeitungen finden dann nur noch in der zusammengenommenen Dokumentation statt.

10.4.22. Produktqualität

10.4.22.1. Functionality

Die Funktionalität des Produktes richtet sich nach dem Dokument *Anforderungsspezifikation.pdf* und ist auf das Dokument *Domainanalyse.pdf* gestützt. Änderungen, Erweiterungen oder Einschränkungen der Funktionalität am Programmcode werden erst dann vorgenommen, wenn die Funktionsänderung säuberlich und eindeutig in den entsprechenden Dokumenten beschrieben wurde. Änderungen in den Dokumenten werden unverzüglich an alle Mitglieder kommuniziert.

Die entwickelten Funktionen werden, je nach Komplexitätsgrad, von mindestens einem weiteren Mitglied kurz auf Richtigkeit und Konformität überprüft. Fehler oder Unschönheiten werden gleich selber korrigiert oder dem zuständigen Entwickler kommuniziert.

10.4.22.2. Usability

Das Produkt, was ein zu integrierendes Modul des Frameworks ist, wird in das Framework integriert. Das erstellte Modul kann mit den bestehenden Modulen über definierte Schnittstellen kommunizieren.

Das Modul kann alleine oder mit anderen Modulen kombiniert eingesetzt werden.

Die Eingabe von Informationen wird so weit wie möglich auf ein Minimum reduziert. Mehrfache Eingaben der gleichen Information sind gänzlich zu vermeiden. Die Ausgabe beinhaltet maximal den Informationsgehalt, welche der Benutzer braucht. Der Aufbau wird so gestaltet, dass sich Änderungen am Informationsgehalt leicht vornehmen lassen.

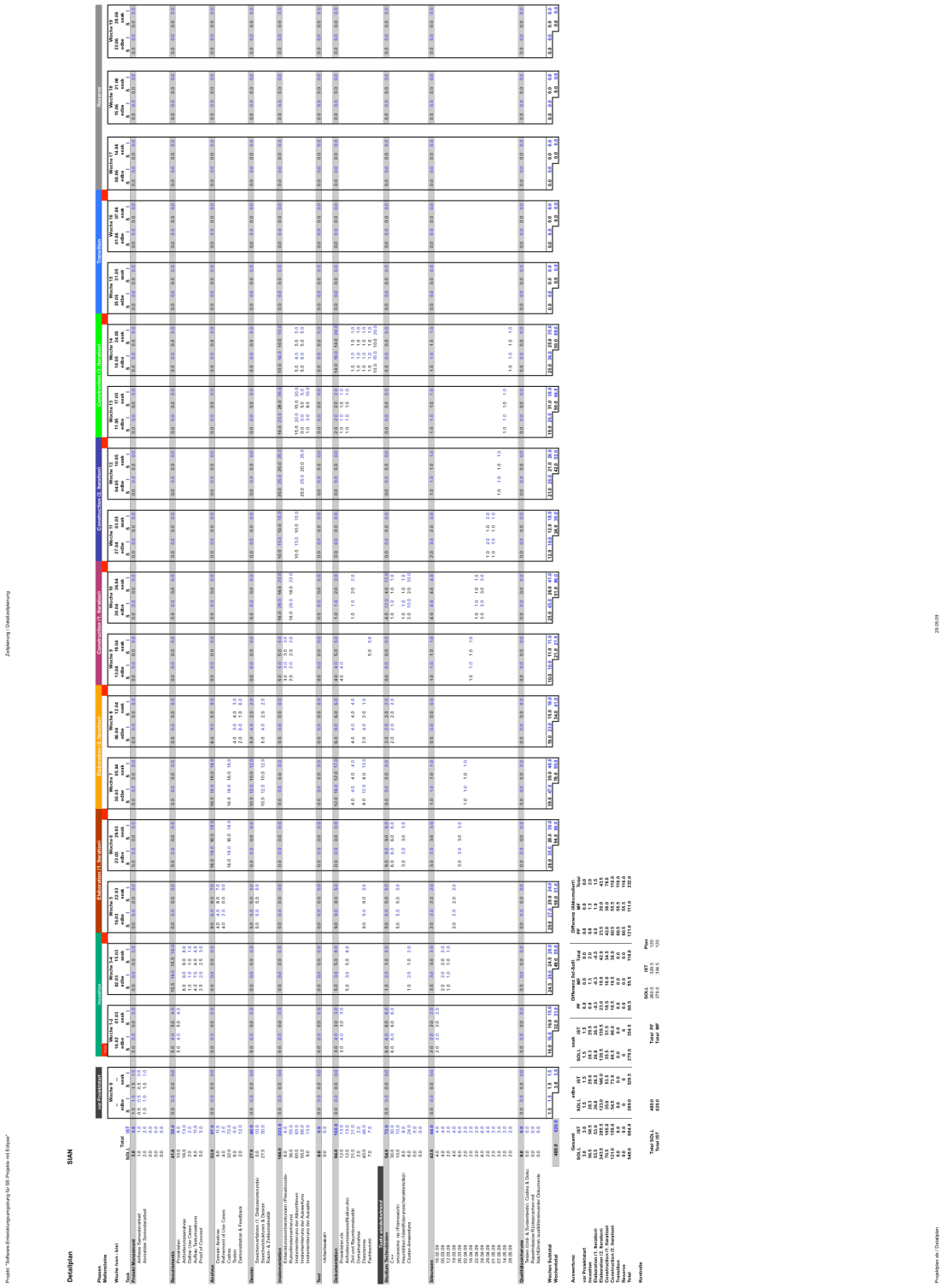
10.4.22.3. Reliability

Durch geeignete Unit- und Usertests wird die Fehlertoleranz des Systems erhöht. Fehleingaben durch den Benutzer sind zu testen. Das Verhalten bei Fehlerzuständen wird bei der Entwicklung getestet. Die Systemdaten sind in jeder erdenklichen und nichterdenklichen Situation wiederherzustellen. Serverseitig werden alle Änderungen protokolliert.

10.4.22.4. Performance

Antwort- und Verarbeitungszeiten des Systems sind in der *Anforderungsspezifikation.pdf* zu finden. Der Ressourcenverbrauch spielt für uns eine primäre Rolle, da sehr grosse Datenmengen analysiert und verarbeitet werden.

10.5. Zeitplan



10.6. System-Testdokumentation

10.6.1. Voraussetzungen

Voraussetzung für diese Tests ist, dass der Prototyp kompiliert und ausgeführt werden kann. Dass die Library und der Cluster der ETH zur Verfügung steht. Auch müssen die Testdaten auf dem Cluster vorbereitet werden. Die Voraussetzungen bis auf fehlerfreie Ausführung (Runtime-Errors) sind erfüllt.

10.6.2. Vorbereitungen

Auf dem Cluster der ETH muss ein Node frei sein, sodass wir unseren Prototyp dort ausführen können. Auch müssen die Testdaten auf dem Cluster vorbereitet sein, sodass diese verarbeitet werden können. Auch muss der Benutzer in diesem Pfad, wo die Output-Datei geschrieben wird, über Schreibrechte verfügen.

10.6.3. Systemtest

In diesem Test werden die Regeln einzeln getestet.

Use Case	*Implementiert*	*Logik überprüft?*	*Fehler / Unschönheiten*	*Status*
H1	Ja	Ja	Runtime Error	NOK
H2	Ja	Ja	Runtime Error	NOK
H3	Ja	Ja	Runtime Error	NOK
H4	Ja	Ja	Runtime Error	NOK
F1	Ja	Ja	Runtime Error	NOK
F2	Ja	Ja	Runtime Error	NOK
F3	Ja	Ja	Runtime Error	NOK
F4	Ja	Ja	Runtime Error	NOK
F5	Ja	Ja	Runtime Error	NOK
F6	Ja	Ja	Runtime Error	NOK
F7	Ja	Ja	Runtime Error	NOK
F8	Ja	Ja	Runtime Error	NOK
F9	Ja	Ja	Runtime Error	NOK
F10	Ja	Ja	Runtime Error	NOK

10.6.4. Verbesserungsmöglichkeiten

10.6.4.1. Bekannte Einschränkungen

Es wurden zwar erfolgreich einige Runtime Errors beseitigt doch sind immer noch 2, 3 Errors vorhanden, sodass wir wegen mangelnder Zeit die Test nicht zum OK führen konnten.

10.6.4.2. Mögliche Detailverbesserungen

Der erste Schritt wäre sicher mal alle Runtime Errors zu beseitigen. Sobald dies geschehen ist, können die Tests nochmals durchgeführt werden und die Regeln sollten alle auf OK wechseln.

10.7. Sitzungsprotokolle

Sitzungsprotokoll

Projekt:	**Software zur Identifikation von Applikationsklassen in NetFlow-Daten**
Woche:	**1. Woche**
Datum:	**Montag, den 16.02.09**

Teammitglieder / Kürzel:

Bernhard Tellenbach
Eduard Glatz

Edon Berisha
Selim Akyol

Traktanden:

- Vorstellung des Projekts
- Einführung
- Fragestellung

Diskussion / Beschlüsse:

-Dokumentation und Einführung des bestehenden Software-Framework

-Identifikation von Applikationsklassen
(Auswertung und Analyse der Andwenundsprotokolle?)

-Top-N-Analyse oder Zeitanalyse

-Programm das File einliest oder Rechner am Router mit Lifestream angeschlossen ist.
(konkret online oder offline Analyse)

-Art des File, Format etc.

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Erstellung eines kleines Programm, dass bestimmte Felder aus einem NetFlow Packet ausliest.	Seak Edbe
Login bereitstellen	

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstag, den 05.03.09

Zeit: 16:30

Dauer: ca. 2h

Sitzungsprotokoll

Projekt: **SIAN**
Woche: **Woche 3**
Datum: **Donnerstag, den 05.03.08**

Teammitglieder / Kürzel:

Selim Akyol
Edon Berisha
Eduard Glatz
Bernhard Tellenbach

Traktanden:

- Besprechung der Anforderungsspezifikation
- Besprechung des Projektplans
- Einführung in das NetFlow Projekt (Crash-Kurs)
- Projektstatus besprechen
- Weiteres Vorgehen besprechen

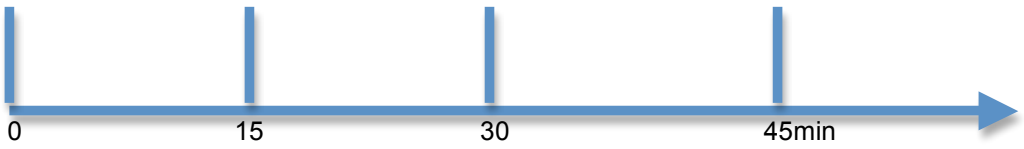
Diskussion / Beschlüsse:

Erkenntnisse der Sitzung vom 05.03.09

Allgemeine Infos:

Der Datenstrom wird in 15min-grossen Zeitschlitzen aufgeteilt. (oder 5min)

Zur Analyse sind immer genau 3 Teile zu beachten (=45min).



Kommt ein Packet im 3. Zeitschlitz an, kann die Auswertung vom 1. Zeitschlitz abgeschlossen werden.

AS = Autonomes System

AS 559 ist Switch(CH) alle anderen AS ist Ausland.

NetFlow-Record:

Ein NetFlow-Record entspricht einer Verbindung. Das heisst in einem NetFlow-Record befinden sich mehrere UPD, TCP etc, Pakete.

<srIP, dstIP, srcPort, destPort, TOS, Protocol>

Mehrere Records zusammen mit einem Header werden als ein UPD Packet vom Router an den Backup-Server geschickt.

Definierbare Parameter: (Modul soll parametrisierbar sein)

-Ob Zeitschlitze 5min oder 15min betragen

-Start und Endzeit

Output des Moduls:

-CSV-Datei (möglicher Output)

Datum, Web(in%), P2P(%), Attacks(%)

01.01.09 12:30-12:45, 50%, 30%, 10%

Eine Stunde NetFlow-Packete darf Max. 1 Stunde Real-Time verarbeitung kosten.

Mögliche Speicherung der Daten im Modul:

<SourceIP, SourcePort> →HashMap

<DestinationIP> →HashMap

<DestPorts> →Array

Endpunkt = IP & Port (Source oder Destination)

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
CSV	Seak
NetFlow	Edbe
Zeitschlitze	edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstag, den 12.03.09

Zeit: 15:00

Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: Woche 4
Datum: Donnerstag, den 12.03.09

Teammitglieder / Kürzel:

Eduard Glatz

Edon Berisha

Selim Akyol

Traktanden:

- Besprechung Projektplan
- Projektstatus besprechen
- Demonstration Codemässig des Proof of Concept Prototyp
-
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

- Aufzeigen was für Daten für jede Regel gespeichert wird.

- Aufzeigen von möglichen Speicherstrukturen.

- Zielanforderung: 12 Mio Flows (10min) in weniger als 10min abarbeiten.

- Aufpassen betreffend Source und Destination, Target und Initiator

- Anwendung der Heuristikregeln ist H1-H5, F1-F9 (siehe Anforderungsspezifikation.doc)

- Ein NetFlow ist unidirektional (in 1 Richtung) und beinhaltet mehrere "Real"-Pakete (TCP/UDP/etc.)

- Immer 3 Intervalle speichern aufgrund der Zeitüberschneidung, damit keine Informationen verloren gehen.

- 1 Intervall = 10 Minuten.

- "1. Intervall" ist abgeschlossen, wenn ein Paket geparst wird, das zum entsprechend "3. Intervall" gehört.

- Router exportiert alle 15 Minuten NetFlow-Daten.

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Anfspez gemäss Feedback bearbeiten	Seak
Aufzeigen von Möglichen Datenstrukturen	Seak / Edbe
Aufzeigen welche Daten Relevant sind pro Regel	Edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstag, den 19.03.09

Zeit: 16:30

Dauer: -

Sitzungsprotokoll

Projekt: SIAN
Woche: Woche 5
Datum: Donnerstag, den 19.03.09

Teammitglieder / Kürzel:

-seak
-edbe
-Eduard Glatz
-Bernhard Tellenbach

Traktanden:

- Besprechung der Speicherarten
- Besprechung des Proof of Concept Beispielprogramm
- Besprechung Projektplan-Feedback
- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

-P2P-Ports / Non-P2P-Ports →Tabelle 3 (CP-Sheet)
-Globale FlowList (Globale Liste aller NetFlows)
-Hash-Map in Cluster-Link (Doku von Herrn Tellenbach)
-gleicher Flow → overwrite Counter hinzufügen
-Bitmaske vordefinieren (für Appl. Klassen)
-uint32 für TCP=1, UPD=2 Beide = 3
1. Regel → P2P Flows (H1-H5)
2. Regel → F1 – F10
-jeder Flow geht durch alle Regel
-Anz Flow & % Anteil als Output (CSV)
-1. Zeikomplexität wichtib bei Detail-HM
2. Raumkomplexität
Pseudocode mit den Regeln
Alle Regel sind anwendbar

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Regel anpassen	Edbe
Pseudocode	Edbe seak
Zeit und raumkomplexität anschauen	Seak

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstag, den 26.03.09

Zeit: 15:00

Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: Woche 6
Datum: Donnerstag , den 26.03.09

Teammitglieder / Kürzel:

Eduard Glatz
Bernhard Tellenbach
Selim Akyol
Edon Berisha

Traktanden:

- Pseudocode Präsentation
- Pseudocode Besprechung
- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

-Erweiterung des Pseudocode:
F9 hat gleiche Daten wie H4 → kann gespart werden
F10 und H5 müssen auch implementiert werden
Bitmaske anstatt Counter

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Pseudocode erweiterungen	Seak / edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstg, den 02.04.09

Zeit: 15:00

Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: Woche 7
Datum: Donnerstag, den 2.04.09

Teammitglieder / Kürzel:

seak
edbe
Eduard Glatz

Traktanden:

- Projektstatus besprechen
- Fragen besprechen
- Ergebnisse aufzeigen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

-Malware-Ports nicht zeitlos → F9 Attacken (s.124)
-Internet sicherheit-Thema

-CP-Sheet Prio 1. (Andere nur Ports-Tabellen anschauen)

-Unverschlüsselte Verbindungen → kein Problem

-RYPE (AS-Nummern), Cablecom, Bluewin, Switch

-Aurara (IMB – Research)

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Pseudocode prototyp	Seak edbe
F9 Malware analyse	Edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstag, den 16.04.09
Zeit: 15:00
Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: Woche 8
Datum: Donnerstag, den 16.04.09

Teammitglieder / Kürzel:

Eduard Glatz

Bernhard Tellenbach

seak

edbe

Traktanden:

- Besprechung Pseudocode (2.Disskusionsrunde)
- Besprechung Projektdokumentation
- Projektstatus besprechen
- Offene Fragen
- Weiteres Vorgehen besprechen

Fragen:

-In BitMaske bei F5 Regel in welcher Reihenfolge? Von links nach Rechts oder von Rechts nach links?

-Sweep / Scan / Dos oder ob nur F9?

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Erweiterung Pseudocode	Seak / edbe
Bitmasken analyse	edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Mittwoch, den 22.04.09

Zeit: 15:00

Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: 10
Datum: Mittwoch, den 22.04.09

Teammitglieder / Kürzel:

seak
edbe
Bernhard Tellenbach
Eduard Glatz

Traktanden:

- Sitzungsprotokoll der letzten Sitzung besprechen
- IP V4 oder V6 werden nicht unterschieden
- Bitmaske Array besprechen
- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

-Formel für Module Rechnung impl.
-BitmaskArray + Tempbitmaske zum setzen der einzelnen Bitmasken
-defaultwerte beachten
-Code sparen mittels einmaliger überprüfung
-null prüfen
-H3 als funktion

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Modulo Rechnung	Edbe
Bitmaskarray	Seak
Code sparen	Seak
Refactoring	edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Freitag, den 29.04.09

Zeit: 15:00

Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: Woche 11
Datum: Mittwoch, den 29.04.09

Teammitglieder / Kürzel:

seak
edbe
Eduard Glatz

Traktanden:

- Dokumentation allgemein besprechen
- Klassendiagramm besprechen
- Flussdiagramm besprechen
- Zeit & Raumkomplexität
- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

-2 Ansätze der Intervalle anschauen und studieren
-HashMap im leeren Zustand ca. 1,5Kb
-Abklären ob UnitTest nötig
-Pointer nicht als Referenzen weitergeben!
-SSD erstellen

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
SSD	Edbe
Tests	Seak
Hasmap	Seak
Intervalle	Ebe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Freitag, den 01.05.09

Zeit: 15:00

Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: Woche 11
Datum: Freitag, den 01.05.09

Teammitglieder / Kürzel:

Seak
edbe
Bernhard Tellenbach

Traktanden:

- Sitzungsprotokoll der letzten Sitzung besprechen
- Projektstatus besprechen
- Bitmasken auswertung
- Switch-Case Fall bei Auswertung
- Counter Problem
- Bei mehreren F Regeln
- H1-H4 P2P Regeln zusammenfassen?
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

-H-Regeln in ein Array zusammenfassen
-F10 Regeln wenn keine H oder F1-F9 Regeln ausgeschlagen hat
-Cluster-> -ulimit auf 2GB Ram beschränken
-Cluster→ ssh x02 (node wechseln)
-Interval gemäss Tellenbach vorschlag anpassen

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Regeln anpassen	Edbe
Interval anpassungen	seak

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstag, den 07.05.09
Zeit: 15:00
Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: 12
Datum: Donnerstag, den 07.05.09

Teammitglieder / Kürzel:

seak
edbe
Eduard Glatz
Bernhard Tellenbach

Traktanden:

- Sitzungsprotokoll der letzten Sitzung besprechen
- Projektstatus besprechen
- Prototyp besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

-H3 Problem mit der Sortierung→Sortier Funktion nötig
-keine GlobaleFlowliste mehr sondern Interval bezogene Liste
-in 10Min → ca. 20Mio Flows->worst case 40Mio IP's
-Anzahl EndPoint = 1/10 von Flows
-Code mittels Oxygen kommentieren

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Sortierfunktion schreiben	Seak
Intervalliste	edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstag, den 14.05.09
Zeit: 15:00
Dauer: 1h

Sitzungsprotokoll

Projekt: SIAN
Woche: Woche 13
Datum: Donnerstag, den 14.05.09

Teammitglieder / Kürzel:

seak
edbe
Eduard Glatz
Bernhard Tellenbach

Traktanden:

- Sitzungsprotokoll der letzten Sitzung besprechen
- Projektstatus besprechen
- Prototyp besprechen
- Fachbericht besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

-Im Fachbericht Sweep und Scann unterscheidung erklären
-Zeitschlitzintervall + 30 Minuten inkl. Lücken müssen gedeckt werden
-Division durch Null beachten
-HostDataObject keine Klassenvariablen sondern Referenzen benutzen
-F2->Unterscheidung zwischen scrHostDataObject und dstHostDataObject
-Iteratoren lokal definieren

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Doku	Seak
Zeitschlitzintervall	Seak
HostDataObject src und dst	Edbe
Iteratoren	Edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Donnerstag, den 28.05.09

Zeit: 15:00

Dauer: 2h

Sitzungsprotokoll

Projekt: SIAN
Woche: 15
Datum: Donnerstag, den 28.05.09

Teammitglieder / Kürzel:

Eduard Glatz
Bernhard Tellenbach
Seak
Edbe

Traktanden:

- Sitzungsprotokoll der letzten Sitzung besprechen
- Projektstatus besprechen
- Letzte Anpassungen an das Projekt
- Abschluss besprechen

Diskussion / Beschlüsse:

-Code erweiterungen gemäss Tellenbach betreffend der HashFunktion
-Code Doku mittels Oxygen generiert
-Abgabe auf CD unbedingt Library mitgeben.
-Ressourcenerwähnung der 3 wichtigtsen Dokumenten
-Bei Pseudocode originaltext miteinbeziehen.
-Letzte Anpassungen an Dok

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

Was	**Wer**
Letzte Anpassungen an Dok	Seak
Letzte anpassungen an Code	edbe

Kommende Abwesenheiten

Wer	Von	Bis	Bemerkung

Nächster Termin:

Datum: Freitag, den 29.05.09
Zeit: 16:00
Dauer: 1h