Leopold-Franzens-Universität
Innsbruck

Institut für Informatik

Dokumentation
Erweiterung des Programms ‚extended ping'

Bakkalaureatsarbeit

eingereicht bei Dr. Ing. Michael Welzl

Marcus Fischer

Innsbruck, 04. Juli 2004

Abstract:

Um die Differenz von Slow Path (Verarbeitung in Software) und Fast Path (Verarbeitung in Hardware) von Routern im Internet zu prüfen, wurde ein Programm entwickelt, welches mittels einer Erweiterung von Ping-Paketen (ICMPs) Zeitunterschiede errechnet. Das Programm stützt sich dabei auf die Möglichkeit, bei einem Ping-Paket zusätzlich Optionen mitzuführen. Als Pings mit Optionen bei dieser Untersuchung verarbeitet wurden, hat sich herausgestellt, dass häufig beim Host Pakete mit Optionen nicht ankamen.

In dieser Arbeit wird nun mit Hilfe einer Erweiterung des entsprechenden Programms untersucht, welche bzw. wie viele Router und Hosts in der Lage sind, Pakete mit Optionen zu verarbeiten. Bei einem Test mit rund 17.700 Hosts hat sich als Resultat herausgestellt, dass insgesamt 6,2% der Pakete mit Optionen nicht beantwortet wurden. Dabei konnten 63,6% der Router auf dem Hinweg, 21,6% der Router auf dem Rückweg bzw., der Host selbst und 14,8% der Router mit unbekannter Ursache die Pakete nicht verarbeiten.

Inhaltsverzeichnis

1 Vorbetrachtung

1.1 Einleitung

Die vorliegende Ausarbeitung schließt an die Arbeit „On the Impact of IP Option Processing“[1] von Mattia Rossi und Michael Welzl an. Dies ist die Erweiterung für das in der oben genannten Arbeit beobachtete Problem der fehlenden Antworten auf Pings mit gesetzten Optionen.

Während den Auswertungen der Daten stellten Rossi und Welzl fest, dass häufig Hosts auf Pings mit gesetzter Option nicht reagierten. Da solche sogenannten NOP-Fehler in großer Anzahl erscheinen, ergibt sich die Fragestellung, an welcher Stelle genau die Pakete mit gesetzten Optionen verworfen werden: Bei dem Host selbst oder bei einem oder mehreren Routern auf dem Weg zum Host. Hier soll nun genau diese Frage beantwortet werden. Dazu wird eine Erweiterung des in der oben genannten Arbeit verwendeten Programms „extended ping“ vorgestellt, welche die zentrale Einheit dieser Ausarbeitung darstellt.

In der Erweiterung des ursprünglichen Programms wird mit Hilfe einer Analyse des Pfades bei einem fehlgeschlagenem NOP-Ping-Versuch, der Router bzw. Host ausfindig gemacht, welcher Optionen nicht weiterleitet bzw. nicht verarbeiten kann.

Im Folgenden wird zunächst die Problematik der NOP-Fehler vorgestellt. Anschließend wird der Erweiterungsvorschlag wie auch die Entwicklung des entsprechenden Algorithmus detailliert dargestellt. Nach der Erläuterung zum Versuchsaufbau und kurzen Bemerkungen zur Durchführung werden die Ergebnisse präsentiert.

[1] Rossi , Welzl: *On the Impact of IP Option Processing*, Preprint-Reihe des Fachbereichs Mathematik-Informatik, No.15, University of Innsbruck, Oct. 2003.

1.2 Definitionen

1.2.1 Host

Ein Host (engl. ‚Empfänger') ist der entsprechende Rechner, den man auf irgendeine Weise erreichen möchte. Im Kontext dieser Arbeit ist der Host der Zielrechner des Pfades, der untersucht werden soll.

1.2.2 Ping

Ping (Packet InterNet Groper) ist ein softwarebasiertes Werkzeug um die Erreichbarkeit eines Hosts zu überprüfen. Dazu werden von dem Programm sogenannte ICMP Echo-Requests an den zu erreichenden Host gesendet. Auf diese muss der Host laut Spezifikation des Pingprotokolls ein ICMP Echo-Reply zurückgeben. Die Zeit, die während dem Senden des Requests (Anfrage) und dem Empfangen des ICMP Echo-Replys (Antwort) verstreicht, ist die sogenannte round-trip time. Diese gibt Auskunft über die Distanz und die Stauverhältnisse, die auf der Strecke zum Host liegen.

1.2.3 Traceroute

Traceroute ist ein softwarebasiertes Werkzeug mit dem der gesamte Datenweg zwischen zwei Rechnern, die über ein Netzwerk verbunden sind, angezeigt werden kann. Traceroute listet dabei alle Router auf, die an der Datenübertragung zwischen dem Traceroute Aufrufer und der Zieladresse beteiligt sind.

1.2.4 Trace

Ein Trace ist eine Liste von Routern, die am Ende eines Traceroutes entsteht, der sogenannte Pfad.

1.2.5 Optionen, IP-Optionen

Ein IP-Paket kann bis zu 40Byte an (IP-)Optionen beinhalten. Die in der RFC 791[2] spezifizierte Option ‚NOP - No Option' wird für das Trennen zweier Optionen verwendet und hat, wie der Name schon darauf hinweist, keine wirkliche Funktion.

1.2.6 NOP und NONE

Das Programm ‚extended ping' versendet abwechselnd ICMP-Echo-Request Pakete (Ping Pakete) mit gesetzer ‚no option' IP-Option (=NOP) und Pakete mit keinen Optionen (=NONE).

[2] Jon Postel: Internet Protocol Standard RFC 791, STD0005, Internet Engineering Task Force, September 1981

2 Erweiterung zu ‚extended Ping'

2.1 Problematik

In der Arbeit „On the Impact of IP Option Processing" von Rossi und Welzl, sollte mit Hilfe des Programmes ‚extended ping' herausgefunden werden, welche Zeitunterschiede zwischen der Bearbeitung von IP-Paketen mit Optionen und der Bearbeitung von IP-Paketen ohne Optionen bei Routern entstehen. Dabei wurden mittels eines Programms Pings zu einem Host versendet, um Messdaten zu sammeln, mit denen Vergleiche durchgeführt werden können.

Eine unerwartete Nebenbeobachtung bei der Auswertung der dort zur Verfügung stehenden Daten war, dass einige Pings mit der Option ‚no option', kurz NOP, nicht beim Host ankamen.

Bei der letzten Messung [1] mit ‚extended ping' antworteten rund 3.000 Hosts nicht auf Optionen. Dem Nicht-Anworten der Hosts können drei Ursachen zu Grunde liegen, die im Folgenden näher erklärt werden:

2.1.1 Ursache 1

Der Host selbst kann die Information der NOP-Optionen in den IP-Paketen weder interpretieren noch weiterleiten. In diesem Fall sind alle Router in der Lage gewesen, die NOP-Optionen der IP-Pakete zu lesen und zu versenden.

2.1.2 Ursache 2

Ein Router auf dem Weg zum Host ist nicht in der Lage, die NOP-Option eines IP-Paketes zu lesen oder zu verarbeiten. Dieses Paket wird dann nicht weitergeleitet.

2.1.3 Ursache 3

Da der Rückweg vom Host ein anderer sein kann, als der Hinweg, kann die Situation eintreten, dass ein Router auf dem Weg zurück die NOP-Option eines IP-Paktetes nicht lesen oder verarbeiten kann, auch wenn auf dem Hinweg alle Router das IP-Paket mit NOP-Option verarbeiten konnten.

2.2 Erweiterungsvorschlag: Nachverfolgung von Routern

Um bei jedem fehlgeschlagenem NOP-Ping die entsprechende Ursache herauszufinden, ist es notwendig, jeden Router des Pfades auf seine Fähigkeit, NOP-Pakete verarbeiten zu können, zu überprüfen.

2.2.1 Bemerkung

Es muss beachtet werden, dass, sobald ein Router gefunden wurde, der die IP-Pakete nicht weitersendet, für alle nachfolgenden Router bzw. für den Host keine Aussage mehr getroffen werden kann, ob IP-Pakete mit NOP-Option von ihnen verarbeitet werden können oder nicht.

In der hier vorgeschlagenen Erweiterung, werden nun sämtliche Router bzw. der Host auf ihre Fähigkeit, NOP-Pakete zu verarbeiten, überprüft.

Dabei muss eine Reihe von Annahmen getroffen werden:

2.2.2 Annahme 1

Ein Router, der nicht auf Pings mit Option reagiert, aber die Pakete mit Option korrekt weiterleitet, gilt als ein Router, der NOP-Pakete verarbeiten kann.

2.2.3 Annahme 2

Es wird angenommen, dass der Pfad sich während der Analyse des Hosts nicht ändert.

2.3 Algorithmus

Um herauszufinden, an welcher Stelle genau der erste Fehler in dem Pfad auftritt, wurde ein spezieller Algorithmus ausgearbeitet. Dieser Algorithmus springt immer genau dann ein, wenn ein Fehler bei der Weitergabe des Paketes festgestellt wird. Er kann zusätzlich zu dem Programm ‚extended Ping' mit Hilfe eines speziellen Parameters aufgerufen werden. In diesem Abschnitt wird zunächst eine für den Algorithmus entwickelte Klassifikation vorgestellt und anschließend auf den Algorithmus selbst eingegangen. Dabei wird zuerst ein Überblick über den Ablauf des Algorithmus gegeben, um danach einzelne Besonderheiten detailliert zu veranschaulichen.

2.3.1 Router-Status Klassifikation

Der im folgenden Absatz vorgestellte Algorithmus benötigt eine eindeutige Klassifikation der einzelnen Router bezüglich ihrer Fähigkeit, NOP-Pakete zu verarbeiten. Aus diesem Grund wurde eine spezielle Router-Status Klassifikation ausgearbeitet. Hier werden die Router drei unterschiedlichen Klassen zugeordnet. Je nachdem, ob ein Router auf Pings mit und ohne NOP-Optionen antwortet, nur auf Pings ohne Optionen, oder ob ein Router im Pfad hinter einem nicht-antwortendem Router liegt und somit keiner der beiden anderen Klassen eindeutig zugeordnet werden kann, werden die Klassifizierungen den Routern vergeben. Im Folgenden werden die drei Klassen aufgeführt:

Y = Router antwortet auf Pings mit und ohne NOP-Optionen.
N = Router antwortet nur auf Pings ohne Option.
X = Router antwortet auf keine Pakete oder liegt im Pfad hinter einem mit ‚N' klassifizierten Router. Die mit ‚X' klassifizierten Router können somit theoretisch fähig sein, Optionen zu verarbeiten. ‚X' steht daher für ‚unbekannt'.

2.3.2 Algorithmus

Hier wird eine vereinfachte Darstellung des entwickelten Algorithmus vorgestellt. Eine detaillierte Übersicht der angewandten Funktionen im Algorithmus sowie der Veränderungen im Main-Programm befindet sich in Kapitel 5.

Algorithmus (vereinfacht):

1. Beim Auftreten einer ‚NOP ping timeout'-Fehlermeldung, welche besagt, dass auf einen Ping mit NOP-Option keine Antwort empfangen wurde, wird der Algorithmus aktiviert.

2. Er führt zunächst einen Traceroute zum Host durch, der auf das Paket ohne Optionen keine Antwort zurückgegeben hat. Ist der Traceroute nicht vollständig, so wird der aktuelle Hosts verworfen und zum nächsten Host übergegangen.

3. Falls der Traceroute erfolgreich ist, wird jeder Router des Pfades, beginnend mit dem ersten Router des Traces, einmal mit und einmal ohne Optionen angepingt.

4. Jeder Router wird entsprechend seiner Fähigkeiten bzw. Status des Vorgängerrouters mit Hilfe der Router-Status Klassifikation eingestuft.

5. Ist der Pfad zum Host komplett durchgelaufen, nimmt das Programm den normalen Ablauf wieder auf.

Siehe dazu auch die Darstellung des Algorithmus in Abbildung 1.

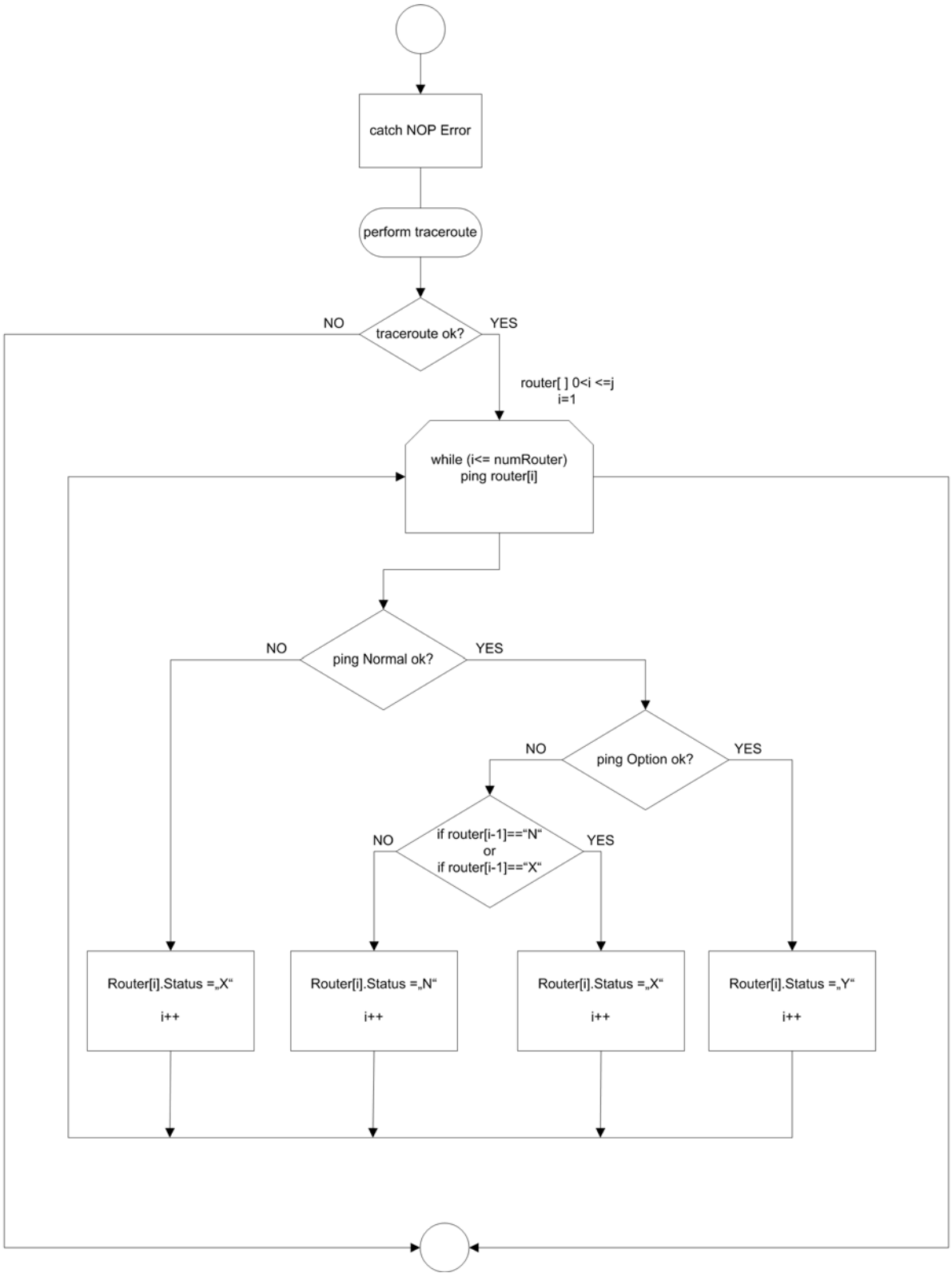

Abbildung 1 Vereinfachte Darstellung des Algorithmus

2.3.3 Besonderheiten

Für den oben gezeigten Algorithmus werden zwei Listen verwendet, um notwendige Informationen während des Ablaufs zu sammeln. Die Liste ‚currentpath' wird während dem Ausführen des Algorithmus parallel zum Abarbeiten aller Router des Pfades aufgebaut. Dabei wird zu jedem gerade verarbeiteten Router die Klassifikation in die Liste abgelegt. Die Liste ‚globallist' wird nach einem kompletten Durchlauf eines Pfades mit Hilfe des Inhalts der ‚currentpath'-Liste ergänzt. Dabei werden gegebenenfalls die Klassifikationen der bereits in der Liste enthaltenen Router überschrieben. Die Ordnung X<N<Y wird dabei fest eingehalten. Das heißt, mit ‚X' gekennzeichnete unbekannte Router können mit ‚N' oder ‚Y' überschrieben werden. Falls ein mit ‚N' gekennzeichneter Router sich als funktionstüchtig erwiesen hat, kann dieser mit ‚Y' überschrieben werden (eventuell ist dieser über eine andere Route geprüft worden). Noch nicht in der globalen Liste enthaltene Router werden neu aufgenommen.

Jeder ‚currentpath' wird vor dem Einpflegen auf Abweichungen überprüft. Die Funktion ‚analysepath' korrigiert die Klassifikationen des gesamten Pfades in folgendem Fall: Auf einen Router mit Status ‚N' folgt ein Router mit Status ‚Y'. Dies würde heißen, dass zwar der Router mit dem Status ‚N' nicht auf Optionen reagiert, diese aber dennoch verarbeitet. Der Fehler wird damit behoben, dass die gesamte ‚currentpath'-Liste, beim letzten Router beginnend, bis zum ersten überprüft wird. Beim Auftreten des ersten Routers mit dem Status ‚Y' werden alle übrigen Router, die im Pfad vor diesem Router liegen, ebenfalls mit ‚Y' klassifiziert.

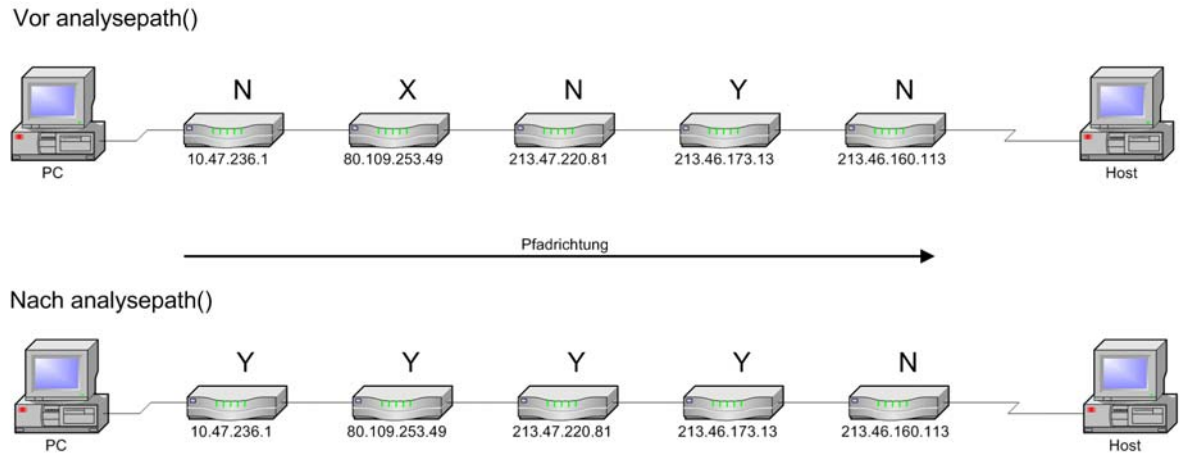

Abbildung 2 Korrektur durch die Funktion ‚analysepath'.

Bei dem Aufbau der ‚currentpath' – Liste wird vor dem Überprüfen der einzelnen Router im Pfad die ‚globallist' untersucht, ob der jeweilige Router zuvor schon klassifiziert wurde. Ist der Status des Routers ‚Y', so wird dieser automatisch ohne weitere Überprüfung in die ‚currentpath' – Liste übernommen. Router, die mit ‚X' oder ‚N' klassifiziert worden sind, wie auch noch nicht klassifizierte Router, werden nochmals überprüft.

Während dem Ablauf des Programms werden zusätzlich zu den fünf Dateien von ‚extended ping' die zwei Dateien routerdiscover.dat und globallist.dat erzeugt. Die Datei routerdiscover.dat enthält alle Router bzw. den Pfad zu einem Host mit der jeweiligen Klassifizierung. Die gesammelten Daten aus der Liste ‚globallist' und Statistiken über die einzelnen Router werden in die Datei globallist.dat geschrieben.

```
GL    1 ip:      192.168.0.1  Status: Y
GL    2 ip:       10.47.236.1  Status: Y
GL    3 ip:    80.109.253.49  Status: Y
GL    4 ip:    213.47.220.81  Status: Y
GL    5 ip:   213.46.173.229  Status: Y
GL    6 ip:    213.46.173.13  Status: Y
GL    7 ip:   213.46.160.113  Status: Y
GL    8 ip:    213.46.161.57  Status: Y
(…)
GL 1972 ip:    170.224.248.7  Status: X
GL 1973 ip:     12.122.2.245  Status: Y
GL 1974 ip:     12.124.34.38  Status: Y
GL 1975 ip:      17.112.4.11  Status: X
GL 1976 ip:       66.28.6.10  Status: Y
GL 1977 ip:     66.250.7.138  Status: Y
GL 1978 ip:     171.64.1.152  Status: X
GL 1979 ip:     171.64.1.167  Status: X
GL 1980 ip:     171.64.1.177  Status: X

Stats:
Routers with Status
    N   |    X   |    Y
  207   |  608   | 1165

Total Number of Hosts checked:  804

Number of Routererrors:         574
Number of Hosterrors:           115
Number of Unknownerrors:        115
```

Auszug aus der Datei ‚globallist.dat'

```
||=============================================================================||
||                  ||                                                         ||
||  NUMBER:      2   ||  HOSTNAME:  12.100.16.103                              ||
||                  ||                                                         ||
||=============================================================================||

    192.168.0.1|     10.47.236.1|   80.109.253.49|   (…)   |    66.93.12.120|
      Y        |       Y        |       Y        |   (…)   |       Y        |
HOST ERROR
```

Auszug aus der Datei ‚routerdiscover.dat' (Die Länge des Traces wurde gekürzt)

2.4 Implementierung des Algorithmus

Das ursprüngliche Programm ‚extended ping' wurde in der Programmiersprache C für Linux implementiert. Dabei wurde eine modulare Struktur verwendet. Um das Programm auszuführen benötigt der Benutzer Root-Rechte, da es einen RAW-Socket erzeugt. Das Programm wird aus der Kommandozeile gestartet.

Der erweiternde Algorithmus wurde zusätzlich in das Programm eingefügt. Der Aufruf des Programms erweitert sich somit um den Parameter ‚-d' (Discover) und dem Dateinamen, unter dem die Ergebnisse gespeichert werden sollen.

Exemplarischer Programmaufruf:

```
ext_ping –i hosts.dat –o results.dat –p 3 –d routerresults.dat
```

3 Resultate

3.1 Definitionen zum Test

3.1.1 Umfang und Dauer

Aus einer Liste von rund 140.000 Hosts wird eine Teilmenge für den Test herangezogen. Die Dauer des Tests ist auf vier Tage festgelegt. Nach Ablauf dieser Testperiode wird das Programm abgebrochen und alle übrigen Hosts der Liste verworfen. Im Test wird jeder Host bis zu dreimal mit bzw. ohne Optionen angepingt; zwischen jedem Ping wird eine Sekunde gewartet.

3.1.2 Hardwarevoraussetzungen

Der Test wird über eine Internetverbindung des Service-Providers Chello über Kabel-Fernsehen mit einer Bandbreite von 1Mbit Download und 128kbit Upload durchgeführt. Das Programm ‘extended ping’ wird auf einem AMD Rechner mit 800Mhz Prozessor und 512MB Arbeitsspeicher mit dem Betriebssystem Linux (Kernel 2.4.20 Gentoo) ausgeführt.

3.2 Auswertung und Ergebnisse

Die Testsession dauerte vier Tage. In dieser Zeit wurden 17.673 Hosts von der modifizierten Version des Programms ‚extended ping' verarbeitet. Bei 12.298 Hosts traten schwerwiegende Fehler auf (zBsp. keine Antwort auf NONE oder NOP Pings), bei den restlichen 5.375 Hosts antworteten 4.282 auf Pings mit Optionen; 1.093 antworteten nicht auf Pings mit Optionen.

Klassifikation	Anzahl Hosts	Prozent
None & NOP	4.282	24,2%
None	1.093	6,2%
Fehler	12.298	69,6%
Gesamt	**17.673**	**100%**

Tabelle 1 Auflistung der Ergebnisse

3.2.1 Auswertung der Fehlerdatei

Von den 12.298 Einträgen in der Fehlerdatei (error.dat) waren 7.392 Hosts nicht ansprechbar, das heißt, sie antworteten auf keine Art von Pings. 2.067 der Hosts antworteten zwar auf Pings mit Optionen, die entsprechenden Daten waren allerdings aufgrund eines fehlgeschlagenen Traceroutes für die weitere Analyse des Pfades nicht verwendbar.

Fehler	Anzahl Hosts	Prozent
NONE Traceroute timeout	2.076	16,9%
NOP Traceroute timeout	2.067	16,8%
NONE und NOP Ping Timeout	7.392	60,1%
Andere Fehler	763	6,2%
Gesamt	**12.298**	**100%**

Tabelle 2 Auswertung der Fehlerdatei

3.2.2 Auswertung der ‚globallist' Datei

Klassifikation	Anzahl Router	Prozent
Y	1730	62,3%
N	264	9,5%
X	784	28,2%
Gesamt	**2.778**	**100%**

Tabelle 3 Auswertung der einzelnen Router mit Status

Die in der Datei ‚globallist.dat' gesammelten Informationen des Testlaufs sind summiert in Tabelle 3 dargestellt. Auf den 1.093 analysierten Pfaden wurden insgesamt 2.778 verschiedene Router identifiziert und entsprechend klassifiziert.

Die folgende Tabelle zeigt das Ergebnis nach der Analyse des jeweiligen Pfades.

Fehlertyp	Anzahl	Prozent
Router	695	63,6%
Host	236	21,6%
Unbekannt	162	14,8%
Gesamt	**1.093**	**100%**

Tabelle 4 Ergebnis nach dem Ausführen des Algorithmus

Von den 1.093 analysierten Pfaden zu Hosts, lag es zu 63,6% an den Routern, dass die Pings mit Optionen nicht weitergeleitet wurden. 21,6% der nicht weitergeleiteten Pakete mit Optionen sind auf die Hosts selbst und 14,8% auf bisher unbekannte Ursachen zurückzuführen.

264 Router waren dafür verantwortlich, dass insgesamt 695 Pakete mit Optionen den Host nicht erreichen konnten. Dies deutet darauf hin, dass einige Router Optionen nicht verarbeiten können oder so konfiguriert sind, dass sie die Pakete mit Optionen nicht weiterleiten.

Beispielsweise ist der Router mit der IP 137.164.23.5 die Ursache für fehlgeschlagene NOP-Ping Versuche an insgesamt sieben Hosts. Siehe dazu auch die Skizze in Abbildung 3.

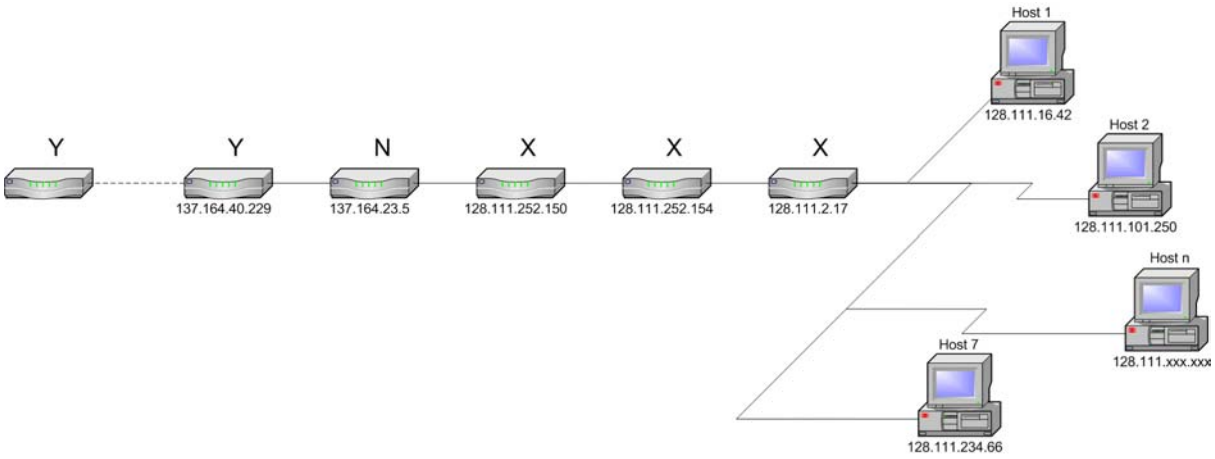

Abbildung 3 Beispiel eines Fehlers

Bei den Host-Fehlern (236) könnte man darauf schließen, dass möglicherweise eine Firewall die Pakete abhält, oder aber ein anderer, nicht Optionen-freundlicher Rückweg die Pakete nicht weiterleitet. Dies bleibt noch zu erforschen.

3.3 Zusammenfassung

Ziel dieser Arbeit war es, herausfinden, an welchen Stellen im Pfad Pakete mit Optionen verworfen werden. Hierzu konnten die Beobachtungen und das Programm ‚extended ping' der Arbeit ‚On the Impact of IP Option Processing' [1] herangezogen werden. Da das Programm ‚extended ping' über keinerlei Analysefunktionen zu Pfaden verfügte, wurde zunächst eine Erweiterung des bestehenden Programms vorgenommen. Um eine Funktion zur Pfadanalyse erweitert, konnten mit dem Programm ‚extended ping' diejenigen Rechner bestimmt werden, die Pakete mit Optionen verwerfen.

Nachdem die Testdaten verarbeitet wurden, konnten die resultierenden Ergebnisse ausgewertet werden. Insgesamt wurden die Pakete mit Optionen bei der Mehrheit der Router und Hosts weitergeleitet. Von den relevanten 5.375 Hosts antworteten 79,7% auf Pings mit Optionen. Dabei wurde festgestellt, dass ein sehr großer Anteil von 69,6% der insgesamt

17.673 untersuchten Hosts, Pings aufgrund von unvollständigen Traceroutes, ungültigen Adressen und ähnlichem nicht verarbeitet wurden.

Die Gruppe der Router, welche Pings mit Optionen nicht weiterleitete, nimmt dabei 63,6% der nicht beantworteten Pings mit Optionen ein. Als Ursache kann hier möglich sein, dass die Router entweder explizit konfiguriert wurden, um solche Pakete zu verwerfen, oder aufgrund ihrer technischen Begebenheit solche Pakete nicht verarbeiten können. Hier waren 9,5% der Router für insgesamt 63,6% der nicht weitergeleiteten Pakete verantwortlich. Der Grund für die recht hohe Anzahl von Routerfehlern und die gleichzeitig geringe Anzahl nicht funktionierender Router liegt darin, dass ein einzelner Router im Netzwerk an mehrere Rechner verlinkt sein kann. Wenn ein solcher Router Pakete mit Optionen nicht verarbeitet, wird bei jedem Pfad, auf dem dieser Router liegt, der Ping mit Optionen ab hier nicht mehr weitergesendet.

Die Gruppe der Hosts, welche Pings mit Optionen nicht weiterleiteten, einschließlich der Router, die auf einem Rückweg die Pakete nicht weiterleiten, nimmt 21,6% der verworfenen Pings ein. Hier kann vermutet werden, dass die Hosts hinter einer Firewall liegen, der Host die Pakete mit Optionen nicht verarbeiten kann oder auf dem Rückweg ein oder mehrere Router die Pakete mit Optionen nicht interpretieren konnten.

Die übrigen 14,8% der verworfenen Pings nimmt die Gruppe der unbekannten Ursachen ein.

Es konnte in dieser Arbeit gezeigt werden, dass in den meisten Fällen von nicht beantworteten Pings mit gesetzten Optionen nicht die Hosts, sondern die Router auf dem Pfad Pakete mit Optionen nicht verarbeiten konnten. Offen bleibt die Analyse der Rückwege vom Host zum aufrufenden Rechner. Ist hier ein Ping nicht weitergeleitet worden, so wurde der entsprechende Host als ‚Fehlerquelle' gekennzeichnet.

3.4 Aufgetretene Probleme und Hinweise

Die Liste der zu untersuchenden Hosts enthält 140.000 Einträge. Es wurden insgesamt sechs Pings, jeweils drei mit und drei ohne Optionen versendet. Nach jedem abgesetzten Ping wird eine Sekunde gewartet. Dieser Vorgang gilt auch für das Überprüfen der einzelnen Router im Pfad. Somit ergab sich im Test eine durchschnittliche Verarbeitungsgeschwindigkeit von ca. vier Hosts pro Minute. Aus diesem Grund konnte nur ein Teil der Hostliste im Rahmen dieser Arbeit untersucht werden.

4 Literaturverzeichnis

[1] Rossi, Welzl: On the Impact of IP Option Processing, Preprint-Reihe des Fachbereichs Mathematik-Informatik, No.15, University of Innsbruck, Oct. 2003.

[2] Jon Postel, Internet Protocol Standard RFC 791, STD0005, Internet Engineering Task Force, September 1981.

5 ANHANG

5.1 Änderungen am Sourcecode des Programms ‚extended ping'

5.1.1 Neuentwickelte Routinen und Funktionen

Die Datei ‚llist.c' enthält alle Funktionen um doppelt verzeigerte Listen aufzubauen und zu bearbeiten. Jedes Element der Listen (globallist und currentpath) enthält jeweils ein Char-Array, welches die IP-Adresse trägt, und ein einzelnes Char-Symbol, welches den Status des Routers trägt. Es gibt je Liste zwei Zeiger, welche auf den Anfang sowie das Ende der Liste zeigen.

Im Folgenden werden die verwendeten Funktionen beschrieben:

`llunit* findInGL(char*):`

Die Funktion **`findInGL`** durchsucht die globale Router-Liste nach einem Router und gibt bei Erfolg einen Zeiger auf das gefundene Objekt zurück. Wird kein Router gefunden, so wird ein NULL-Zeiger zurückgegeben.

Eingabeparameter: IP-Adresse des Routers (char*)

Rückgabeparameter: Zeiger (llunit*) auf den gefundenen Router oder NULL bei Misserfolg.

`void addtoGlist(char*, char*):`

Die Funktion **`addtoGlist`** erzeugt ein neues Routerelement (llunit) mit der übergebenen IP-Adresse und dem Status des Routers und fügt diese in die ‚currentpath'-Liste ein.

Eingabeparameter: IP-Adresse (char*), Status (char*)

Rückgabeparameter: keine

`void addtoCPlist(char*, char*):`

Die Funktion **`addtoCPlist`** erzeugt ein neues Routerelement (llunit) mit der übergebenen IP-Adresse und dem Status des Routers und fügt diese dann in die ‚globallist'-Liste ein.

Eingabeparameter: IP-Adresse (char*), Status (char*)

Rückgabeparameter: keine

`void mergeCP2GL():`

Die Funktion **`mergeCP2GL`** fügt der Liste ‚globallist' die fertiggestellte 'currentpath'-Liste an. Falls der einzufügende Router in der ‚globallist' bereits enthalten ist, wird der Status des Routers gegebenenfalls angepasst.

Eingabeparameter: keine

Rückgabeparameter: keine

`void cleanup():`

Die Funktion **`cleanup`** wird dazu verwendet, den nicht mehr benötigten, zuvor reservierten Speicher freizugeben.

Eingabeparameter: keine

Rückgabeparameter: keine

`void analysepath():`

Die Funktion **`analysepath`** geht einen fertig durchlaufenen Pfad vom Host zum Startpunkt des Pfades ab. Dabei überprüft und korrigiert die Funktion die gesetzten Klassifikationen.

Eingabeparameter: keine

Rückgabeparameter: keine

`int checkStatus(llunit*, char*):`

Die Funktion **`checkStatus`** vergleicht die Klassifikation eines gegeben Routers des Typs (llunit) mit einem gegeben Status (char). Bei Gleichheit wird der Integerwert 1, sonst der Wert 0 zurückgegeben.

Eingabeparameter: Routerelement (llunit*), zu überprüfende Status (char*)

Rückgabeparameter: Integerwerte 0 und 1

`int checkLastCPStatus(char*):`

Die Funktion **`checkLastCPStatus`** kontrolliert, ob der letzte Router in der 'currentpath'-Liste den übergebenen Status besitzt und gibt bei Erfolg eine 0 zurück.

Eingabeparameter: zu überprüfende Status (char*)

Rückgabeparameter: gibt die Rückgabe des strncomp zurück.

`int checkcplastNull():`

Die Funktion **`checkcplastNull`** prüft, ob die ‚globallist' leer ist. Da es sich hier um eine doppelt verkettete Liste handelt, ist dies gegeben, wenn der Zeiger auf das letzte Element dieser Liste auf NULL zeigt. Ist dies der Fall, so wird der Wert 1, in allen anderen Fällen wird die Zahl 0 zurückgegeben .

Eingabeparameter: Routerelement (llunit*), zu überprüfende Status (char*)

Rückgabeparameter: Integerwerte 0 und 1

`int printCPStatus(FILE *in):`

Die Funktion **`printCPStatus`** schreibt den aktuellen Pfad in die als Parameter übergebene Datei. Als Rückgabewert erhält man ein Kürzel für den Typ von Rechner, der das Paket mit Optionen nicht weitergeleitet hat. Dabei steht 1: für Router, 2: für unbekannter Rechner und 3: für Host.

Eingabeparameter: : Zeiger auf den Filehandle in die die Ergebnisse geschrieben werden sollen.

Rückgabeparameter: Integerwert: 1 (= Router), 2 (= unbekannter Rechner), 3 (= Host).

`void printstats(FILE *in):`

Die Funktion **`printstats`** schreibt die zusammengetragenen Ergebnisse der einzelnen Router in die angegebene Datei.

Eingabeparameter: Zeiger auf den Filehandle, in welchen die Ergebnisse geschrieben werden sollen.

Rückgabeparameter: keine

5.1.2 Änderungen an der Datei main.c

Ab Zeile 211 wurden in der Datei main.c rund 200 Zeilen Code eingefügt. Mit Hilfe der oben beschriebenen Routinen ist der Algorithmus aus Kapitel 2 umgesetzt worden. Das Main-Programm wird zusammen mit den Übergabeparametern: Hostliste, Anzahl der Pings, Anzahl der abzuhandelnden Hosts, Name der Ergebnisdateien aufgerufen. Taucht während dem Abarbeiten der Hosts die Fehlermeldung ‚Ping Timeout (NOP)' auf, so wird der ergänzte Algorithmus aufgerufen:

Codeauszug Zeile 214

```
if (Err.flag && strncmp("ping_loop",Err.level,9)==0 && strcmp("Ping Timeout
(NOP)",Err.message)==0)
```

Mit dieser Anweisung wird der Fehler abgefangen und als erster Schritt die Funktion **`trace_main`** aufgerufen. Während die Funktion **`trace_main`** den entsprechenden Pfad zum Host nachvollzieht, wird als überprüft, ob es innerhalb diesen Pfades ein ‚Traceroute Timeout'-Fehler gibt. Tritt an einer Stelle ein Fehler auf, so wird dieser in die Fehlerdatei übernommen. Ansonsten wird jeder Router, der in der Traceroute enthalten ist, in der globalen Routerliste der Reihe nach mit der Funktion **`findInGL`** gesucht.

Codeauszug Zeile 269

```
if(diffrenthost == 0)
search=findInGL(router[j]);
```

Jeder Router, der in der globalen Routerliste mit dem Status ‚Y' gefunden wird, wird direkt mit der Funktion **`addtoCPlist`** in die Current-Path-Liste übernommen.

Codeauszug Zeile 277

```
if (diffrenthost == 0 && search != NULL && checkStatus(search, "Y") )
{
      printf("\n Router in List with Status Y\n");
      addtoCPlist(router[j], "Y");
}
```

Sobald ein Router der Traceroute in der globalen Routerliste gar nicht oder mit einer der Klassifizierungen ‚N' oder ‚X' gefunden wird, so wird dieser und jeder nachfolgende Router der Traceroute nacheinander angepingt.

Codeauszug Zeile 294

```
/* ping the router */
ping_main(router[j]);
```

Konnte der Ping mit Optionen zu einem Router ohne Fehler durchgeführt werden, wird auch dieser Router in die Current-Path-Liste mit der Funktion **`addtoCPlist`** übernommen. Er erhält dabei die Klassifikation ‚Y'.

Codeauszug Zeile 297

```
if (!Err.flag)
{
      printf("\nrouter works ->Y\n");
      addtoCPlist(router[j], "Y");
}
```

Konnte zu einem Router kein Ping erfolgreich durchgeführt werden, wird dieser Router mit der Klassifikation ‚X' an die Current-Path-Liste ebenfalls mit der Funktion **`addtoCPList`** angehängt.

Auch Router, die lediglich Pings ohne Optionen empfangen haben, werden mit Hilfe der Funktion **`addtoCPlist`** an die Current-Path-Liste angehängt. Hier muss allerdings vorher überprüft werden, ob der vorangehende Router mit einem ‚Y' klassifiziert wurde. In diesem Fall erhält der Router die Klassifikation ‚N', in allen anderen Fällen wird der Router mit ‚X' klassifiziert.

Sobald alle Router eines Pfades verarbeitet worden sind, wird die Funktion **`analysepath`** aufgerufen, welche die oben beschriebene Analyse der Current-Path-Liste durchführt.

Nach der Analyse wird die Current-Path-Liste auf die Router untersucht, die Pings mit Optionen nicht weiterleiten konnten.

Anschließend wird mit der Funktion **mergeCP2GL** die Liste Current-Path in die Globallist eingefügt.

Codeauszug Zeile 350

```
/* merge the Path to the Global list */
printf("\nMerging Lists\n");
mergeCP2GL();
```

In diesem Abschnitt wurde ansatzweise auf den Code eingegangen. Für detaillierte Einzelheiten können die Kommentare im Code direkt weitere Auskunft geben.