Technische Universität München
Lehrstuhl Informatik VIII
Prof. Dr.-Ing. Georg Carle
Dipl.-Ing. Stephan Günther, M.Sc.
Johannes Naab, M.Sc.

Programmieraufgaben zur Vorlesung Grundlagen Rechnernetze und Verteilte Systeme

Programmieraufgabe 2 – Neighbor Discovery (3 Punkte)
(18. Mai – 2. Juni 2015)

Abgabe via SVN bis ~~Sonntag, 31. Mai 2015~~ Dienstag, 2. Juni 2015, 23:59:59.99 Uhr (MESZ)

1 Neighbor Discovery Protocol (NDP)

Die Adressauflösung von Network Layer Adressen zu Link Layer Adressen erfolgt unter IPv4 durch das Address Resolution Protocol (ARP). Bei IPv6 [1] wird diese Funktionalität jedoch durch das Neighbor Discovery Protocol[2] (NDP) bereitgestellt. Der Austausch von Protokoll-Informationen erfolgt über das Internet Control Message Protocol Version 6 (ICMPv6) [3]. Da IPv6 in den nächsten Jahren Version 4 des Internet Protocol ablösen wird, wird in dieser Programmieraufgabe die Adressauflösung im Neighbor Discovery Protocol näher betrachtet. Ziel dieser Progammieraufgabe ist es, eine komplette Adressauflösung zu einem, im *link-lokalen* Netz verfügbaren, Nachbarn durchzuführen. Im Detail soll dazu ein Paket mit einer *Neighbor-Solicitation*-Nachricht erstellt, versendet und anschließend die zugehörige *Neighbor-Advertisement*-Nachricht empfangen, interpretiert und ausgegeben werden. Die IPv6 Ziel-Adresse soll dabei als Kommandozeilenparameter übergeben werden können. Für eine detaillierte Beschreibung der oben genannten Standards, halten Sie sich an die referenzierten RFCs.

Wie in der ersten Programmieraufgabe werden auch dieses Mal wieder Rahmenprogramme für C und Java, über das SVN Repository unter *pub/assignment2/*, zur Verfügung gestellt. Es bietet sich an, die Rahmenprogramme als Grundlage für die Abgabe zu verwenden. Stellen, an denen die Programme modifiziert werden müssen, sind mit einem “TODO” gekennzeichnet. Um Ihnen eine Rückmeldung zu Ihrer Abgabe vor der Deadline zu ermöglichen, können Sie wieder das Test-Framework nutzen, welches die eingecheckte Programmversion auf ihre Funktionsfähigkeit testet. Wenn noch Unklarheiten zur Benutzung von Makefiles und Testumgebung bestehen, halten Sie sich bitte an die Angaben vom Aufgabenblatt *assignment1*.

Wichtig:

- Vergewissern Sie sich, dass sie alle, für die Abgabe relevanten, Dateien auch in SVN eingecheckt haben (`svn add <filename>`), damit Diese bei einem Commit auch übertragen werden.
- (Adaptierte) Code Snippets (z.b. von *Stackoverflow*, Umfang von mehr als einer Zeile) *müssen* mit einer Quelle (URL) versehen sein.
- Die Test-Ergebnisse sind *nicht* die alleinige Bewertungsgrundlage für die Abgaben, sondern stellen lediglich eine Orientierung dar, ob die Abgabe den Erwartungen entspricht. Kriterien, die nicht von dem Test-Framework berücksichtigt werden können sind z.B. die korrekte Berücksichtigung der Byte-Order oder graceful Termination ohne Exceptions. Die Qualität des Source Code wird hingegen sehr wohl bewertet. Eine Erklärung der möglichen Test-Outputs wird innerhalb der nächsten Tage im SVN veröffentlicht.
- Erscheinen Sie *vorbereitet* zur Programmierübung. Sie sollten vor der Übung zumindest das aktuelle Aufgabenblatt gelesen haben. Falls noch Unklarheiten zur Verwendung von SVN oder VM bestehen, halten Sie sich zunächst an Aufgabenblatt *assignment0*.

Um Ihnen die Implementierung und Recherche zu erleichtern, ist im Folgenden eine Übersicht über den Aufbau der verschiedenen Protokoll-Header und Nachrichten gegeben, die für die Lösung der Aufgabe benötigt werden.

[1] `https://tools.ietf.org/html/rfc2460`
[2] `https://tools.ietf.org/html/rfc4861`
[3] `https://tools.ietf.org/html/rfc4443`

Abbildung 1: Ethernet-Header. *Destination MAC* ist die L2-Solicited-Node-Adresse. *Source MAC* ist die MAC-Adresse des Interface, von dem aus der Rahmen versendet wird. *Ethertype* gibt die Art L2-SDU an und ist im Falle von IPv6 `0x86DD` (auf Byte-Order achten!). Im Anschluss folgt die Payload des Frames, im Falle von NDP, der IPv6 Header.

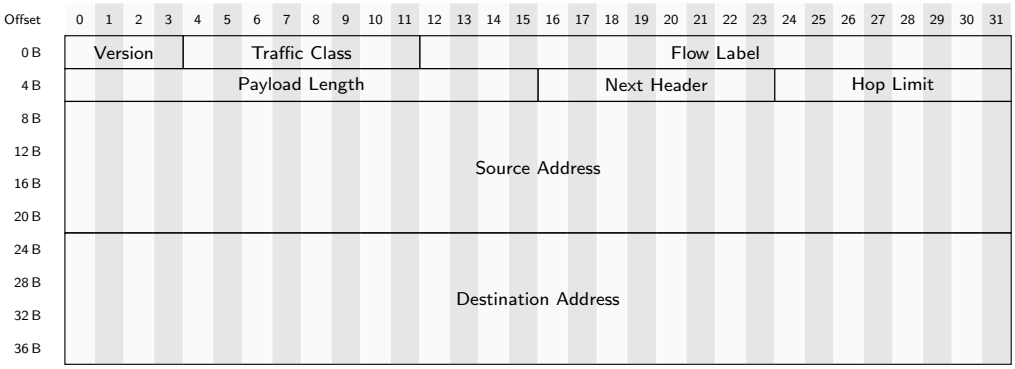

Abbildung 2: IPv6-Header. *Version* bezeichnet die Versionsnummer vom Internet Protokoll (`0x6`), *Traffic Class* ist für NDP `0x00`, ebenso ist das *Flow Label* mit `0` zu füllen (20 bit). *Payload Length* gibt die Länge der Payload in Bytes an. *Next Header* identifiziert den Typ des nächsten (eingekapselten) Headers. Im Falle von NDP enthält das IPv6 Paket eine ICMPv6 Nachricht vom Header-Typ `0x3a`. *Hop Limit* ist das IPv6 Equivalent zur TTL von IPv4 und wird standardmäßig auf 255 gesetzt. *Source Address* und *Destination Address* entsprechen den IPv6 Sender- und Ziel-Adressen. Gefolgt werden die Felder von Extension Headers oder gekapselten Payload, im Falle von NDP, der ICMPv6 Nachricht.

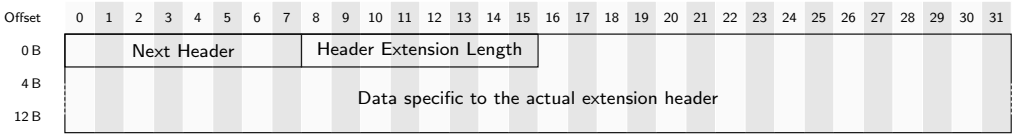

Abbildung 3: IPv6 Extension-Header. IPv6 Pakete können mehrere Extension Header besitzen. Der Typ eines solchen Headers wird im *Next Header* Feld vorangehenden IPv6-Headers bzw. Extension-Headers definiert. Im Extension-Header selbst, wird erneut der Typ des nachfolgenden Headers (*Next Header*), sowie die Länge der Extension (*Header Extension Length)* definiert. Ein Next-Header Feld mit Wert `0x3a` zeigt eine nachfolgenden ICMPv6 Payload an.

(Auf der nächsten Seite geht es weiter.)

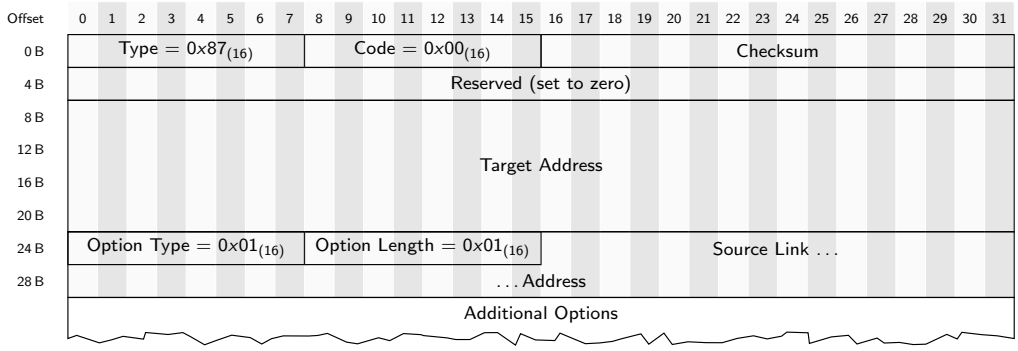

Abbildung 4: Neighbor-Solicitation-Message. *Type* identifiziert die ICMPv6 Nachricht als *Neighbor-Solicitation-Message*, falls `0x87`. Sowohl *Code* als auch *Reserved* werden mit `0x00` initialisiert. *Target Address* identifiziert die IPv6 Adresse, die in eine Link-Layer Adresse aufgelöst werden soll. *Option*: Der Neighbor-Solicitation hängt eine ICMPv6 Option mit Type `0x01` (Source Link Address) und der Länge `0x01` (Vielfache von 8 B einschließlich des ICMPv6 Option Headers). Diese Option enthält als Payload die MAC Adresse des Senders.

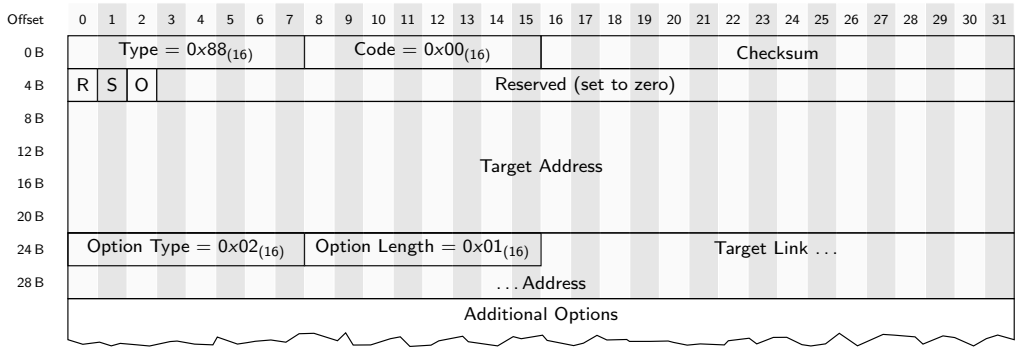

Abbildung 5: Neighbor-Advertisement-Message. *Type* identifiziert das ICMPv6 Paket als *Neighbor-Advertisement-Message*, falls `0x88`. Sowohl *Code* als auch *Reserved* werden mit `0x00` initalisiert. Auf die *Checksum* folgen 3 Flags von jeweils einem Bit, in der Reihenfolge *R,S,O*. *R* ist 1, wenn diese Antwort von einem Router stammt. In betrachteten Fall ist dieses Flag nicht weiter zu berücksichtigen. *S* ist 1, wenn diese Antwort aufgrund einer Unicast-Neighbor-Solicitation Nachricht gesendet wurde. *O* signalisiert, dass die gecachte Link-Layer Adresse geupdated werden soll, und muss im Fall des betrachteteten Szenarios gesetzt sein. *Target Address* enthält die angefrage IPv6 Adresse. *Option*: Dem Neighbor-Advertisement hängt eine ICMPv6 Option mit Type `0x02` (Target Link Address) und der Länge `0x01` (Vielfache von 8 B einschließlich des ICMPv6 Option Headers). Diese Option enthält als Payload die MAC Adresse zu der in *Target Address* gegeben IPv6 Adresse. Das Neighbor-Advertisement kann weitere Optionen in beliebiger Reihenfolge enthalten. Unbekannte ICMPv6 Optionen sind zu ignorieren.

Die *Neighbor-Solicitation-Message* wird von der eigenen MAC und IPv6-Adresse an die *Solicited-Node Multicastadresse* der aufzulösenden IPv6 gesendet. Der Empfänger auf Schicht 2 ist die zur Solicited-Node Multicastadresse gehörende Ethernet Multicastadresse. Die *Neighbor-Advertisement-Message* wird dann per Unicast an den Absender der *Neighbor-Solicitation* gesendet. Hierbei verwendet der sendende Konten als Absender Adresse seine eigenen Adressen auf Schicht 2 resp. Schicht 3.
Für *Neighbor-Solicitation* und *Neighbor-Advertisement* sind für das Hoplimit `0xff` vorgeschrieben. Sollte einem empfange Nachricht hiervon abweichen, so darf diese für die Neighbor Discovery nicht weiter verarbeitet werden.
Es ist auch zu berücksichtigen, dass der IPv6 Header zusätzliche Extension Header und das *Neighbor-Advertisement* zusätzliche Optionen in beliebiger Reihenfolge enthalten kann. Weiterhin sollte Ihr Programm mit dem Empfang von fehlerhaften bzw. unvollständigen Paketen sinnvoll umgehen können.

2 Benutzung der Programmierumgebung

Nach dem Auschecken ihres SVN-Repositories erhalten Sie nun die folgende Ordnerstruktur.

```
assignment2/
|-- C
|   |-- libraw
|   |   |-- include
|   |   |   |-- checksums.h
|   |   |   |-- hexdump.h
|   |   |   `-- raw.h
|   |   |-- Makefile
|   |   `-- src
|   |       |-- checksum.c
|   |       |-- hexdump.c
|   |       |-- raw.c
|   |       `-- timespec.h
|   |-- Makefile
|   `-- src
|       |-- assignment2.c
|       |-- ndisc.c
|       `-- ndisc.h
`-- java
    |-- deps
    ...
```

```
...
 |   |-- GRNVS_RAW.c
 |   |-- GRNVS_RAW.h
 |   |-- libraw
 |   |   |-- include
 |   |   |   |-- checksums.h
 |   |   |   |-- hexdump.h
 |   |   |   `-- raw.h
 |   |   |-- Makefile
 |   |   `-- src
 |   |       |-- checksum.c
 |   |       |-- hexdump.c
 |   |       |-- raw.c
 |   |       `-- timespec.h
 |   |-- manifest.txt
 |   `-- run
 |-- Makefile
 `-- src
     |-- Arguments.java
     |-- Assignment2.java
     |-- GRNVS_RAW.java
     `-- Timeout.java
```

In den Unterordnern `C` und `Java` finden sich die Umgebungen, die Sie für die Bearbeitung der Aufgabe benötigen. Entscheiden Sie sich für *eine* Programmiersprache und kopieren Sie ***den Inhalt eines dieser Ordner*** in ihr Arbeitsverzeichnis:

`/users/<LRZ-Kennung>/assignment2/abgabe/`

Es sollten sich danach die Ordner `src, deps` und die Datei `Makefile` in `abgabe` befinden. Nachdem das gewünschte Rahmenprogramm nach `abgabe` kopiert wurde, muss dort nun `src/assignment2.c` resp. `src/Assignment2.java` bearbeitet werden.

Wichtig: Die Verzeichnisstruktur muss exakt eingehalten werden. Lediglich Quelltext im Abgabeverzeichnis wird bewertet.

2.1 Ausführung

Nachdem im `abgabe` Ordner `make` ausgeführt wurde finden Sie nun die ausführbare Binary *ndisc* in Ihrem Verzeichnis. Diese kann im Ordner `abgabe` mit dem Befehl `./ndisc -i <interface> -t <timeout in sec> <target ipv6 address>` ausgeführt werden. Der *timeout*, welcher als Ganzzahl zu übergeben ist, spezifiziert die Anzahl an Sekunden, die auf den Empfang einer *Neighbor-Advertisement*-Message gewartet werden sollen.

Als Ziel für die Neighbor-Solicitation bietet sich der Router der VMs an. Die IPv6 Adresse des Routers lässt über das Kommando `ip -6 route` anzeigen:

```
# ip -6 route
2a00:4700:0:3::/64 dev eth0  proto kernel  metric 256  expires 86394sec
fe80::/64 dev eth0  proto kernel  metric 256
default via fe80::225:90ff:fe57:224a dev eth0  proto ra  metric 1024  expires 57sec hoplimit 64
            ^^^^^^^^^^^^^^^^^^^^^^^^^
```

Ohne zusätzliche Parameter wird durch die Rahmenprogramme das Netzwerkinterface `eth0` verwendet. Soll das NDP-Paket über ein anderes Interface versendet werden, so kann dieses über den Switch `-i` angegeben werden. Für das Interface `enp3s0` müsste dann `./ndisc -i enp3s0 -t <timeout> <target ipv6 address>` ausgeführt werden.

2.2 Abgabetests

Wenn Sie der Meinung sind, dass Ihr Programm die Anforderungen erfüllt, können sie die Funktionalität durch die bereitgestellte Testumgebung überprüfen lassen. Im Ordner `assignment2` befindet sich seit dem letzten SVN Update die Datei `test-bitte.txt`. Um einen Test anzufordern, verändern Sie diese Datei und committen sie in das SVN Repository. Führen Sie die Testumgebung erst aus, nachdem Sie das Programm selbst ausreichend getestet haben.

Um die korrekte Ausführung des Testers zu gewährleisten, dürfen die Ordnerstrukturen sowie die Makefile nicht verändert werden. Geben Sie außerdem Logging-/Debugging-Output nur auf **stderr** aus (`fprintf(stderr, ...)` bzw. `System.err.println(...)`) und stellen Sie sicher, dass auf **stdout** nur das erwartete Ergebnis ausgebeben wird.

Sie können nur einen Test alle 6 Stunden ausführen lassen. Falls Sie vor Ablauf der 6 Stunden einen zweiten Test beantragen, wird dieser verzögert, bis die 6 Stunden vergangen sind. Die Testumgebung wird nach dem Commit die Ergebnisse des Tests in den Ordner `results/<revision nr>` schreiben, welchen Sie nach einem `svn update` erhalten.

Status	Beschreibung
Success	Die getestete Abgabe hat den Test erfolgreich bestanden.
Failed	Die getestete Abgabe hat den Test nicht bestanden.
Precondition was not satisfied	Der Test wurde nicht ausgeführt, da ein hierfür vorhergehender notwendiger Test nicht erfolgreich war.

Tabelle 1: Beschreibung der Test Status Codes

Testcase	Beschreibung
Launchable	Programm wurde kompiliert und kann gestartet werden
Sends data on Interface	Das Programm sendet irgendwelche Daten an das übergebene Interface
Rejects invalid address	Ungültige IPv6 Adressen werden erkannt, das Programm terminiert direkt.
Sends correct Ethernet Header	Das gesendete Paket hat den erwarteten Ethernet Header
Sends correct IPv6 Header	Das gesendete Paket hat den erwarteten IPv6 Header
Sends correct ICMPv6 Header	Das gesendete Paket hat den erwarteten ICMPv6 Header.
Sends correct ICMPv6 Payload	Das gesendete Paket hat den erwarteten ICMPv6 Payload.
Sends correct Neighbor Solicitation	Die Neighbor Discovery Nachricht ist vollständig richtig.
Timeout without answer	Das Progamm erkennt den Timeout keine Antwort versendet wurde.
Support timeout switch	Das Programm unterstützt, dass der Timeout über `-i <timeout in sec>` eingestellt werden kann.
Accepts correct Neighbor Advertisement	Das Programm akzeptiert das Neighbor Advertisement und gibt die MAC Adresse richtig aus.
Rejects invalid Ethernet Header	Das Progamm verwirft Pakete mit „falschem“ Ethernet Header.
Rejects invalid IPv6 Header	Das Programm verwirft Pakete mit „falschem“ IPv6 Header.
Handles IPv6 Extension Headers	Das Programm kann korrekt mit IPv6 Extension-Headern umgehen.
Rejects invalid ICMPv6 Header	Das Programm verwirft Pakete mit „falschem“ ICMPv6 Header.
Rejects invalid Neighbor Advertisement	Das Programm verwirft Pakete mit „falschem“ Neighbor Discovery Payload.
Handles Neighbor Discovery options correctly	Das Programm kann korrekt mit NDP Optionen umgehen.

Tabelle 2: Beschreibung der Abgabetests

2.3 Hinweise falls Sie nicht die Rahmenprogramme verwenden

Es ist möglich, die Aufgaben in einer Sprache Ihrer Wahl zu lösen. Es gibt ein paar Dinge, die dabei aber beachtet werden müssen:

- Der Arbeitsaufwand muss vergleichbar sein mit dem der Rahmenprogramme (magic.do_ndisc()) währe **nicht** ok. Falls Sie sich unsicher sind, fragen Sie die Übungsleitung.
- Die Ausführbare Datei **muss** *ndisc* heißen.
- Die Ausführbare Datei **muss** die Parameter der Rahmenprogramme unterstützen: `-i <interface> -t <timeout in sec> <target ipv6 address>`
- Es **muss** eine Makefile existieren. Falls Ihre Lösung nicht kompiliert werden muss, da Sie z.B. eine Scriptsprache verwendet haben, muss dennoch eine Dummy-Makefile vorhanden sein, welche ggf. einfach nichts tut.
- Um Pakete vor dem Test zu installieren müssen die Paketnamen in einer Datei *deps.txt* in dem Ordner *abgabe* stehen.
- Pakete dürfen nur aus Debian Jessie *main* und *contrib* installiert werden.
- Im Falle einer erfolgreichen Adressauflösung soll das Programm auf stdout eine Nachricht im Format „::1 is at 01:02:03:04:05:06“ einschließlich eines finalen Zeilenumbruchs ausgeben.
- Im Falle eines Timeouts (`-t <timeout in sec>, default 5`) soll das Programm auf stdout die Nachricht „Message timed out“ einschließlich eines finalen Zeilenumbruchs ausgegeben.
- Sollte die übergebene Adresse keine IPv6 Adresse sein, so soll das Programm ohne Ausgabe auf stdout direkt terminieren.
- Weitere Nachrichten, Logs, etc. sind nicht auf stdout sondern auf stderr auszugeben.