Universität Koblenz-Landau
Fachbereich 4
Institut für Informatik
Arbeitsgruppe Rechnernetze

Weiterentwicklung und Implementierung des RIP-MTI-Routing-Daemons

Diplomarbeit
zur Erlangung des akademischen Grades
„Diplom-Informatiker“

Mai 2008

Bearbeiter : Frank Bohdanowicz
Matrikelnummer: 202120801

Betreuer : Prof. Dr. Ch. Steigner
Dipl. Inform. H. Dickel

Ehrenwörtliche Erklärung

Hiermit versichere ich, Frank Bohdanowicz, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als den von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

_____

Frank Bohdanowicz

Koblenz, den 27.05.2008

Zusammenfassung

Das Routing Information Protocol (RIP) ist ein Internet-Standard-Routing-Protokoll, das einst mit zu den am meisten eingesetzten Routing-Protokollen in IP-Netzwerken gehörte. Es basiert auf dem sogenannten Distanzvektoralgorithmus und ist in seiner Funktion und seinem Aufbau sehr einfach ausgelegt. Seit jeher leidet es allerdings unter dem sogenannten Counting-to-Infinity (CTI) Problem, bei dem die Erreichbarkeit einer eigentlich ausgefallenen Verbindung zu einem Ziel scheinbar aufrechterhalten wird. Die Distanz zu diesem Ziel wird aufgrund des fortwährenden Austauschs von nicht mehr gültigen Verbindungsinformationen zwischen, in einem Ring geschalteten, RIP-Routern hochgezählt, theoretisch bis ins Unendliche. Dabei entstehen Routingschleifen, die den Netzwerkbetrieb erheblich stören können, da die gesendeten Netzwerkpakete aufgrund der Schleife die selben Router immer wieder passieren und weder an ihr eigentliches Ziel gelangen noch verworfen werden können. Die Gefahr des Auftretens des CTI-Problems schränkt die Einsetzbarkeit von RIP enorm ein. Die Netzwerke, in denen RIP eingesetzt wird, können nicht beliebig wachsen, da die maximale Größe des Netzwerks auf eine relativ kleine Distanz zwischen den Routern begrenzt ist, um die Dauer und die Folgen des CTI-Problems im Falle des Auftretens gering zu halten. Je stärker auch die Topologie eines Netzwerks vermascht ist, um mit zusätzlichen, alternativen Verbindungen Ausfällen entgegenzuwirken, umso stärker steigt auch die Gefahr des Auftretens des CTI-Problems nach einem Ausfall.
Bislang existierten für RIP lediglich Mechanismen, die das Risiko des Auftretens und die Auswirkungen des CTI-Problems verringern, das Problem selbst aber nicht beheben können.
Mit „RIP with minimal topology information“ (RIP-MTI) wurde in der AG Rechnernetze an der Universität Koblenz-Landau eine abwärtskompatible Erweiterung zu RIP geschaffen, die das CTI-Problem zu beheben verspricht. Der RIP-MTI-Algorithmus sammelt zusätzliche Informationen über die Topologie des Netzwerks und nutzt diese, um nach dem Ausfall einer Verbindung richtige Informationen über die Erreichbarkeit von Zielen von falschen Informationen unterscheiden zu können.
In dieser Diplomarbeit wird die Implementierung des RIP-MTI-Algorithmus behandelt. Mit Hilfe der speziell entwickelten RIP-Netzwerk-Testumgebung XTPeer, in der das CTI-Problem kontrolliert provoziert werden kann, wird die Wirksamkeit der Implementierung eines Quagga RIP-MTI-Routers überprüft und entsprechend weiterentwickelt. Dafür wird der RIP-MTI-Algorithmus an die Implementierung des Quagga RIP-Routing-Software, sowie an die der Netzwerk-Testumgebung XTPeer, angepasst.
Diese Diplomarbeit wird vom Autor selbst als fortgeschrittene Zwischenstation eingestuft, vor der Herstellung und Herausgabe der Implementierung einer RIP-MTI-Routing-Software, die auch in produktiven Netzwerken eingesetzt werden könnte.

Schlagworte: RIP, RIP-MTI, Counting-to-Infinity, CTI, Routing-Loop, Distanz-Vektor

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Das Routing Information Protocol (RIP) ist ein Internet-Standard-Routing-Protokoll und gehörte mit zu den am häufigsten verwendeten Routing-Protokollen in IP-Netzwerken. Es basiert auf dem Distanzvektoralgorithmus und gilt als der klassische Vertreter der Familie der Distanzvektorprotokolle. Es ist ein sehr anspruchsloses Routing-Protokoll, das eine einfache Implementierung und eine unkomplizierte Verwaltung erlaubt. Aufgrund des Counting-to-Infinity (CTI) Problems ist RIP in seinen Einsatzmöglichkeiten allerdings sehr stark eingeschränkt. Das CTI-Problem ist Ergebnis einer entstandenen Routingschleife, über die eine scheinbare Erreichbarkeit einer eigentlich ausgefallenen Verbindung aufrechterhalten wird, was den Betrieb des gesamten Netzwerks erheblich beeinträchtigen kann. Die Netzwerke, in denen RIP als Routing-Protokoll eingesetzt ist, sind in ihrer Größe und ihrer Komplexität begrenzt und können nur eingeschränkt erweitert werden. Das CTI-Problem ist mit für die schlechten Konvergenzeigenschaften und für die mangelhafte Skalierbarkeit von RIP-Netzwerken verantwortlich und auch ein Grund dafür, dass RIP zunehmends an Bedeutung als Internet-Routing-Protokoll verliert.
Mit dem RIP-MTI-Algorithmus wurde von A. Schmid in [Sch99], in der Rechnernetze AG an der Universität Koblenz-Landau, eine Aussicht auf Behebung des CTI-Problems in RIP-Netzwerken vorgestellt. Der RIP-MTI-Algorithmus ist eine abwärtskompatible Erweiterung der Standard RIP-Spezifikation aus [STD53]. Die Entstehung einer Routingschleife kann anhand der Distanz der betroffenen Ziele im Netzwerk und der Überprüfung vorhandener Schleifen in der Topologie des Netzwerks erkannt und vermieden werden. Das CTI-Problem wird dadurch verhindert. Struktur und Aufbau der Kommunikation unter den RIP-Routern, also die Eigenschaften des Protokolls selbst, werden dabei nicht verändert, was bedeutet, dass RIP-MTI-Router zusammen mit konventionellen RIP-Routern in einem Netzwerk funktionieren und die RIP-Router nach und nach ersetzt, bzw. erweitert werden könnten.
Auf Basis der Vorschläge aus [Sch99] von A. Schmid, für eine Implementierung des RIP-MTI-Algorithmus, entwickelte T. Kleemann in [Kle01] einen Netzwerksimulator, mit dessen Hilfe er das Verhalten von RIP-MTI-Routern in verschiedenen Topologien simulieren und untersuchen konnte. Dabei stellte er konzeptionelle Schwächen des entworfenen RIP-MTI-Algorithmus fest, die er aber durch entsprechende Veränderungen beheben und somit die Funktionalität des RIP-MTI-Algorithmus bestätigen konnte.
Nach den Vorschlägen von A. Schmid erweiterte T. Koch in [Koc05] den RIP-Routing-Daemon der weit verbreitet eingesetzten Software-Routing-Suite Quagga zu einem RIP-MTI-Routing-Daemon, in dem er den RIP-MTI-Algorithmus entsprechend implementierte. Die Veränderungen aus [Kle01] berücksichtigte er allerdings noch nicht. Diese Implementierung bot nun mittelfristig die Möglichkeit, den RIP-MTI-Algorithmus auch in produktiven Netzwerksystemen einzusetzen. Als Testumgebung für die Implementierung wurde die Netzwerkvirtualisierungssoftware VNUML[1] gewählt, die zumindest aus softwaretechnischer Sicht ein sehr wirklichkeitsnahes Umfeld bietet.
Um kritische Netzwerk-Situationen kontrolliert provozieren zu können, entwickelte D. Pähler in [Päh06] den RIP-XT, eine Möglichkeit der externen Ansteuerung der Quagga RIP-Daemons. Der RIP-XT besteht aus einem zentralen XT-Client mit einer grafischen Oberfläche, von dem aus Kontrollkommandos an die, im Quagga RIP-Daemon implementierten, XT-Server gesendet werden können, um das Verhalten und die Weitergabe von Informationen unter den RIP-Daemons zu kontrollieren. Für eine bessere Analyse der Netzwerk-Situationen und des Verhaltens der Quagga RIP-Daemons, erweiterte S. Lange den RIP-XT in [Lan07] zum XTPeer,

1 VNUML: Virtual Network User Mode Linux (siehe Kapitel 4.2)

mit einer Client/Server Struktur für die zentrale Protokollierung der in den RIP-Daemons vorhandenen Informationen. Um kritische Netzwerk-Situationen einfacher erzeugen und reproduzieren zu können erweiterte T. Keupen in [Keu07] den XTPeer, um eine Möglichkeit der, auch automatischen, Generierung von Testfällen zum Zwecke der besseren Analyse der Funktionalität und Robustheit der weiter zu entwickelnden Implementierung des RIP-MTI-Routing-Daemons.

1.1 Motivation

Im Gegensatz zum konventionellen RIP-Algorithmus, der schon seit Jahrzehnten im Einsatz ist und dessen Stärken und Schwächen hinreichend bekannt sind, ist der RIP-MTI-Algorithmus weitestgehend unbekannt und noch nie in produktiven Netzwerk-Systemen eingesetzt worden. Der RIP-Routing-Daemon der Quagga Routing-Suite stellt hierfür aber eine hervorragende Ausgangsbasis dar, um eine Implementierung des RIP-MTI-Algorithmus im Einsatz gegenüber konventionellen RIP-Routern zu testen und das Verhalten in kritischen Netzwerk-Situationen zu vergleichen. Verschiedenste kritische Netzwerk-Situationen können mit Hilfe der Netzwerk-Testumgebung, bestehend aus VNUML und XTPeer, sehr einfach und flexibel herbeigeführt und untersucht werden.
Dem in [Koc05] entwickelten Quagga RIP-MTI-Routing-Daemon fehlt noch eine leistungsfähigere Anbindung an den XTPeer, um die Routing-Informationen des RIP-MTI-Algorithmus detaillierter protokollieren und somit das Verhalten des Quagga RIP-MTI-Routing-Daemons intensiver analysieren zu können. Die vorhandene Implementierung berücksichtigt außerdem noch nicht die bisher festgestellten Schwächen im Konzept des RIP-MTI-Algorithmus. So kann das CTI-Problem in bestimmten Netzwerk-Situationen und -Topologien nach wie vor auftreten, was in [Lan07], [Keu07] und [Wol06] gezeigt wurde.
Mit Hilfe der Testumgebung kann die Implementierung des RIP-MTI-Routing-Daemons vorangetrieben werden. Auch können einzelnen Varianten des RIP-MTI-Algorithmus unabhängig voneinander implementiert und deren Verhalten und Wirksamkeit im Vergleich zueinander untersucht und überprüft werden.
Ziel dieser Diplomarbeit ist die Weiterentwicklung des RIP-MTI-Routing-Daemons der Quagga Routing-Software-Suite. Mit Hilfe des Steuerungs- und Auswertungswerkzeugs XTPeer und der Netzwerkvirtualisierungssoftware VNUML soll die Wirksamkeit des RIP-MTI-Routing-Daemons untersucht und Lösungsvorschläge zur Weiterentwicklung erarbeitet und implementiert werden.

1.2 Aufbau der Diplomarbeit

Im Kapitel 2 dieser Diplomarbeit wird das „Routing Information Protocol“ (RIP) vorgestellt. Hier ist eine nähere Beschreibung des Protokolls und des Counting-to-Infinity (CTI) Problems zu finden. Im Kapitel 3 folgt eine ausführliche Erläuterung des RIP-MTI-Algorithmus, der an der Universität Koblenz entwickelten Erweiterung von RIP zur Verhinderung des CTI-Problems. Kapitel 4 bietet eine Übersicht über die verwendete und weiterentwickelte Software. Hier wird die Erweiterung der Testumgebung XTPeer und die Möglichkeiten der Konfiguration des RIP-MTI-Routing-Daemons aufgezeigt. In Kapitel 5 folgt eine Beschreibung der Implementierung im Zusammenhang mit einigen Problemsituationen. Hier werden die verschiedenen Varianten des RIP-MTI-Algorithmus anhand von Netzwerk-Simulationen mit Hilfe der Testum-

gebung XTPeer erläutert und ein Überblick, über die entwickelten Vorgehensweisen des RIP-MTI-Algorithmus gegeben, sowie eine Empfehlung, welche Variante sich zu bewähren scheint und weiterentwickelt werden sollte. Kapitel 6 schließlich schließt diese Diplomarbeit mit einem Fazit und einem Ausblick auf die Weiterentwicklung des RIP-MTI-Routing-Daemons ab.

1.3 Terminologie

Um Missverständnisse auszuschließen werden im folgenden einige Begriffe und Symbole, die in dieser Arbeit verwendet werden, erläutert.

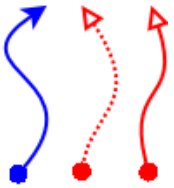
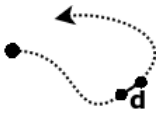
- Subnetz: Als Subnetz wird in IP-Rechnernetzen ein Teilnetz (Kommunikationsteilnetz) bezeichnet, das einen zusammenhängenden Bereich von IP-Adressen aus einem Teil des IP-Adressraums umfasst. Mittels einem als Subnetting bezeichneten Verfahren werden IP-Adressen mit der gleichen IP-Netzwerknummer zu einem Subnetz zusammengefasst. Dies ermöglicht eine bessere Unterteilung der Netz-Klassen (A, B, C) in kleinere Netze (Broadcast-Domains) und mindert das Problem des Adressverlusts. In der Regel werden Subnetze so strukturiert, dass ein hierarchisches Routing vorliegt. Ein Router kann somit mehrere IP-Adressen und auch mehrere Subnetze mit einer IP-Netzwerknummer auf ein einziges Subnetz abbilden.

- Router: Ein Router ist eine Netzwerkkomponente, deren Software es ermöglicht, die Subnetze eines Netzwerks auf der Vermittlungsschicht miteinander zu verbinden, so dass Endgeräte aus den verschiedenen Subnetzen miteinander kommunizieren können. Dabei analysiert der Router ankommende Datenpakete nach ihrer Ziel-IP-Adresse und leitet sie mittels seiner lokalen Weiterleitungstabelle an das entsprechende Subnetz, bzw. an einen Router der näher an dem Subnetz liegt, weiter (die Pakete werden „geroutet“). Die für die Weiterleitungstabellen notwendigen Routing-Informationen tauschen die Router mittels Routing-Protokollen untereinander aus. Ein RIP-Router ist dementsprechend ein Router der das Routing Information Protocol (RIP) verwendet um Routing-Informationen auszutauschen und über den dazugehörigen Algorithmus entsprechend zu verarbeiten.

- Routing Daemon: Mit dem Begriff Daemon wird unter Unix und seinen Derivaten (Linux, BSD,...) im allgemeinen ein Programm bezeichnet, das im Hintergrund abläuft und im System einen bestimmten Dienst zur Verfügung stellt. Daemons werden üblicherweise automatisch mit dem Systemstart geladen. Eine direkte Interaktion mit einem Benutzer findet in der Regel nur zu Konfigurationszwecken statt. In dieser Ausarbeitung wird der Begriff RIP-(MTI-)Daemon (oder auch RIP-(MTI-)Routing-Daemon) mehr im Sinne der softwaretechnischen Realisierung des RIP-Routers, bzw. des RIP-Routing-Dienstes, verwendet und steht für das implementierte Programm, das den Routing-Prozess ermöglicht.

- Interface: Als Interface wird in dieser Ausarbeitung die physikalische Netzwerkschnittstelle zwischen einer Netzwerkkomponente, zum Beispiel einem Router, und dem Netzwerk bezeichnet. Jedem Interface ist hierbei genau eine IP-Adresse zugeordnet, die es auf der Vermittlungsschicht identifiziert.

• Netzwerkschleife: Der Begriff bezieht sich in dieser Arbeit allein auf die physikalische Topologie des Netzwerks und bezeichnet einen Ring aus Routern, der aufgrund gewollter redundanter Verbindungen entstanden ist, um die Ausfall-Sicherheit des Netzwerks zu erhöhen. Diese Ring-Topologie kann Teil einer größeren, vermaschten Netzwerk-Topologie sein. Eine Netzwerkschleife besteht aus mindestens zwei Routern, wenn diese über zwei getrennte, physikalische Leitungen miteinander verbunden sind. Innerhalb von Netzwerkschleifen können auf der Vermittlungsschicht Routingschleifen entstehen, deren Verhinderung Thema dieser Arbeit ist.

Symbol	Erklärung
	RIP-Router / RIP-MTI-Router mit Hervorhebung der Interfaces hier: Router R1 mit den beiden Interfaces A und B
d	(IP-)Subnetz, Verbindung (hier: Subnetz über dem IP-Adressbereich d)
[d | 4 | R1]	Routing-Tabellen Eintrag [Subnetz | Metrik[2] | nächster Router der Route] hier: [Subnetz d | Metrik 4 | Router R1]
d | 4	versendetes Routing-Update / Response-Nachricht (Subnetz | Metrik) hier: Subnetz d | Metrik 4
	Hervorhebung der Verbreitungsrichtung der Routing-Updates mit aktiviertem Split-Horizon[3], was entgegengesetzt der Pfeilrichtung wirkt.
	Hervorhebung der Verbreitungsrichtung der Routing-Updates ohne aktiviertes Split-Horizon. Routing-Updates werden hier in beide Richtungen gesendet
	gültige, ungültige und ausgefallene Route: Der Knubbel am Ende steht für das am Router eintreffende Routing-Update, die Pfeilrichtung zeigt den Verlauf der Route, ausgehend vom Router hin zum Ziel-Subnetz. (entgegen der Verbreitungsrichtung des Routing-Updates) • blau, durchgezogen : gültige Route mit gültiger Metrik • rot, gestrichelt : ungültige Route mit scheinbar gültiger Metrik • rot, durchgezogen : ausgefallene Route mit Metrik RIP-Infinity (16)
	Routen-Kombination, Kombination zweier Routen über dem Subnetz d

Die in den Grafiken verwendeten Symbole

2 Metrik: Distanz zum Ziel-Subnetz über die Route
3 Split-Horizon: Spezielle RIP eigene Filterfunktion für die Weitergabe von Routen, siehe Kapitel 2.3.2

2 Das Routing Information Protocol (RIP)

In diesem Kapitel wird das Routing Information Protocol (RIP) näher beschrieben. Auch das Counting-to-Infinity (CTI) Problem und die Auswirkungen des Auftretens auf das Netzwerk werden hier erläutert.

2.1 RIP Grundlagen

Das Routing Information Protocol (RIP) ist ein sehr einfach ausgelegtes Routing-Protokoll. Es gilt als der klassische Vertreter der Familie der Distanzvektorprotokolle, die nach dem Prinzip „*Teile deinen Nachbarn mit, wie die Welt für dich Aussieht*“ Routing-Informationen austauschen. Distanzvektorprotokolle werden auch häufig Bellman-Ford Protokolle genannt, da sie auf einem Algorithmus der Berechnung des kürzesten Weges basieren, der von R.E. Bellman [Bel57] erstmals beschrieben wurde. Eine erste genaue Beschreibung des verteilten Algorithmus stammt von Ford und Fulkerson [FoFu62]. Routing-Protokolle, die auf dem Distanzvektoralgorithmus basieren, gehörten zu den ersten Netzwerk-Protokollen, die dynamisches Routing in IP-Netzwerken ermöglichten und wurden bereits im ARPANET, das als experimenteller Vorläufer des Internets gilt, eingesetzt. Von dynamischen Routing-Verfahren wird verlangt, dass sie sehr flexibel sind, denn sie sollen sich möglichst schnell an Veränderungen im Netzwerk anpassen, wenn Netzwerk-Knoten oder Verbindungen hinzukommen oder ausfallen. Bei dynamischen Routing-Verfahren liefern verteilte Algorithmen den einzelnen Netzwerk-Knoten Informationen über die Topologie des Netzwerks, die dem Aufbau lokaler Weiterleitungstabellen dienen. Mit Hilfe dieser Tabellen, die beim statischen Routing manuell angelegt werden müssen, finden Datenpakete ihren Weg durch das Netzwerk, vom Sender zum Empfänger. Die Weiterleitung der Datenpakete ist ein einfaches und wohldefiniertes Verfahren, während ein Routing-Algorithmus zur dynamischen Wegewahl unterschiedlichen Konzepten folgen kann und wesentlich komplexer ist.
Beim Internet-Routing existiert eine hierarchische Untergliederung beim Einsatz der Routing-Verfahren zwischen der Klasse der Interior Gateway Protocols (IGP) und der Klasse der Exterior Gateway Protocols (EGP). IGP Protokolle werden innerhalb von Autonomen Systemen (AS) eingesetzt, während EGP-Protokolle außerhalb zum Einsatz kommen. Ein AS ist in diesem Zusammenhang ein Verbund von Routern und Netzwerken, die sich unter einer einzigen administrativen Kontrolle befinden. IGP Routing-Protokolle sind dazu gedacht Routing-Informationen innerhalb eines AS möglichst umfassend zu verbreiten, um den jeweils schnellsten Weg zwischen den einzelnen Netzwerkkomponenten zu finden, während EGP-Protokolle nach eher wirtschafts-politischen Kriterien ausgewählte Routing-Informationen zwischen den AS verbreiten. RIP gehört zur Klasse der IGP-Protokolle. Seine einstmals sehr weite Verbreitung ist hauptsächlich der Tatsache geschuldet, das es zusammen mit der beliebten UNIX-Version der Berkeley Software Distribution (BSD) verteilt wurde, die mit dem Programm „routed“ (route management daemon) eine frühe Version des RIP-Algorithmus enthielt. Im Jahr 1988 wurde dann die erste Version von RIP (RIPv1) als klassenorientierter IP-Routingalgorithmus über das RFC[4] 1058 spezifiziert. Der Nachfolger RIP Version 2 (RIPv2) wurde im Jahr 1993 in den RFCs 1387 bis 1389 und eine erweiterte Version im Jahr 1994 in den RFCs 1721 bis 1724 als klassenlose Form beschrieben, die die jeweilige Präfixlänge der erreichbaren Netze als zusätzliche Information transportiert. Aktuell ist RIPv2 im RFC 2453 aus

4 RFC: Request for Comments: Offene Dokumentenreihe, in der Standards und Empfehlungen zu technischen Entwicklungen und Vorgehensweisen bzgl. des Internets spezifiziert werden. http://www.rfc-editor.org

dem Jahr 1998 beschrieben und unter STD 56 standardisiert. Bei RIP verwaltet jeder Router eine Routing-Tabelle in der jeder Eintrag eine Route ist und aus dem Ziel-Subnetz, der Distanz zum Ziel-Subnetz (Metrik) und dem ersten Nachbarrouter auf dem Weg zum Ziel-Subnetz besteht. Die Distanz basiert bei RIP auf der sogenannten Hop-Count-Metrik und entspricht der Anzahl der Hops, bzw. RIP-Router, auf der Strecke zwischen dem Router und dem Ziel-Subnetz. Jeder RIP-Router versucht immer die Route mit den wenigsten Hops, also mit der besten Metrik, zu einem Ziel-Subnetz zu finden. Nur die Routen mit der besten Metrik zum Ziel-Subnetz speichert der RIP-Router in seiner Routing-Tabelle. Alternative Routen mit einer gleichen oder schlechteren Metrik werden nicht gespeichert, sondern sofort. Bevor der RIP-Router eine empfangene Route in seine Routing-Tabelle einträgt, inkrementiert er deren Metrik um 1 und zählt sich damit selbst mit in die Distanz zum Subnetz mit ein. Fällt eine Verbindung zu einem Subnetz aus, welches von da an für den Router unerreichbar ist, so erhält die eingetragene Route zu diesem Subnetz den höchstmöglichen Metrik-Wert. Diese Metrik wird RIP-Infinity genannt. Sie ist um den Wert 1 höher als die höchste gültige Metrik die eine Route besitzen kann und steht für den Ausfall der Route und die Unerreichbarkeit des Subnetzes. RIP-Infinity wird in der Standard-Spezifikation von RIP über den Wert 16 repräsentiert, was letztlich bedeutet, dass zwischen einem RIP-Router und seinem Ziel-Subnetz maximal 15 RIP-Router, inklusive ihm selbst, liegen dürfen. Dieser relativ kleine Wert für RIP-Infinity erlaubt auch nur kleinen, einfachen Netzwerken mit RIP zu „routen“ und ist den schlechten Konvergenzeigenschaften und damit auch dem, zum Zeitpunkt der Spezifizierung des RIP-Standards, ungelösten CTI-Problems geschuldet. Das Counting-to-Infinity (CTI) Problem verursacht aufgrund einer entstandenen Routing-Schleife ein zyklisches Hochzählen der Metrik einer Route, die eigentlich ausgefallen und deren Subnetz nicht mehr erreichbar ist. Dieses Hochzählen der Metrik kann bislang nicht verhindert oder unterbrochen werden und läuft theoretisch bis in Unendliche. Solange das CTI-Problem abläuft, existiert auch die Routing-Schleife, die zu erheblichen Mehrbelastungen des Netzwerks führen kann, da auch Datenpakete die Router der Schleife wiederholt passieren können und weder verworfen werden noch ihr eigentliches Ziel erreichen.

RIP-Router versenden in regelmäßigen Abständen, über periodische Routing-Updates, den kompletten Inhalt ihrer Routing-Tabelle unaufgefordert an alle direkt benachbarten Router, bei RIPv1 via Broadcast und bei RIPv2 via Multicast. Dementsprechend bilden sie ihre Routing-Tabelle nur über Routing-Informationen aus den, von den unmittelbaren Nachbarroutern, empfangenen Routing-Updates. RIP-Router erhalten Informationen über die Erreichbarkeit der Subnetze somit immer nur aus zweiter Hand, sofern sie nicht selbst direkt mit dem Subnetz verbunden sind.

Für die Kommunikation untereinander verwendet RIP zwei verschiedene Nachrichtentypen, für Anfragen und Antworten, die in [STD53] spezifiziert sind:

1) Request-Nachricht: Mittels der Request-Nachricht sendet ein RIP-Router eine Anfrage an seine benachbarten RIP-Router und fordert sie auf, ihm eine Abbildung des Inhalts ihrer Routing-Tabelle zu zusenden. Für gewöhnlich sendet ein RIP-Router einen Request nur unmittelbar nach dem er gestartet und an das Netzwerk angeschlossen wurde, um möglichst schnell Routing-Informationen zu erhalten. Mit einem Request werden normalerweise alle in der Routing-Tabelle vorhandenen Routen eines RIP-Routers abgefragt. Die RIP-Spezifikation erlaubt jedoch auch einzelne Subnetze explizit im Request anzugeben, über die dann Routing-Informationen eingeholt werden, was aber mehr für Diagnose-Software interessant ist.

2) Response-Nachricht: Die Response-Nachricht ist das Routing-Update, das ein RIP-Router zur Weitergabe von Routing-Informationen versendet. Responses werden vom RIP-Router unaufgefordert und regelmäßig, in einem periodischen Update-Intervall, an alle benachbarten RIP-Router gesendet und enthalten eine Abbildung des kompletten Inhalts der Routing-Tabelle des Routers. Als Triggered-Updates werden Responses bezeichnet, die nach einer Veränderung der Routing-Tabelle versendet werden und nur die Änderungen weitergeben. Mit Responses antwortet der Router auch auf eingehende Requests, um den Inhalt seiner Routing-Tabelle, je nach Anfrage, teilweise oder komplett, dem Anfragenden mitzuteilen.

Die RIP-Nachrichten bauen auf dem verbindungslosen UDP-Protokoll auf. Standardmäßig wird Port 520 für den Austausch der Routing-Updates verwendet.
Eine RIP-Nachricht enthält immer den RIP-Header, der aus einem *command*- und einem *version*-Field von jeweils einem Byte Länge besteht. Das *command*-Field beschreibt, ob die RIP-Nachricht ein Request oder ein Response ist und im *version*-Field wird die verwendete RIP-Version angegeben. Im Nachrichtenformat von RIPv1 sind noch einige Felder als unbenutzt angegeben, die erst mit der Einführung von RIPv2 belegt wurden. Mit jeder RIP-Nachricht können Informationen über maximal 25 Routen versendet werden. RIPv2 unterstützt auch Authentifizierung. Dafür muss allerdings der erste Routen-Eintrag der RIP-Nachricht durch ein Authentifizierungs-Segment ersetzt werden.

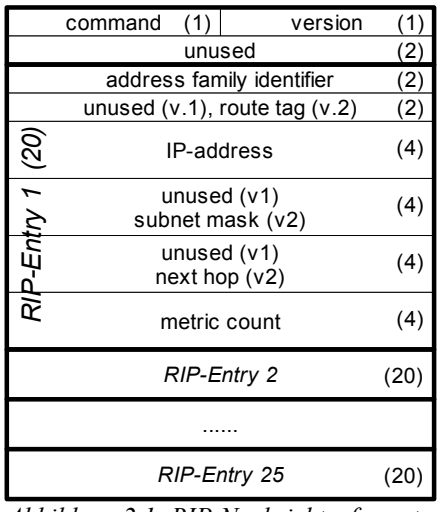


Abbildung 2.1: RIP-Nachrichtenformat

Ein Routen-Eintrag (RIP-Entry) ist 20 Byte lang und besteht aus den folgenden Feldern:

- Das Feld *address family identifier* erlaubt RIP, neben IP, auch andere Addressfamilien zu Unterstützen.
- Das *Route Tag* dient der Unterscheidung zwischen intern (IGP) gelernten Routen und extern (EGP) gelernte Routen, sowie zur Erkennung von RIP-Nachrichten mit Authentifizierung.
- *IP-address* enthält den IP-Adressbereich des Subnetzes, zu dem die Route führt.
- Im Feld s*ubnet mask* ist die Netzmaske des Subnetzes angegeben.
- Über das Feld *next hop* kann die IP-Adresse des nächsten Hops der Route angegeben werden. Dies entspricht für gewöhnlich der IP-Adresse des RIP-Routers, von dem aus das Routing-Update gesendet und die Routing-Information übernommen wurde. (Learned-From-IP-Address)
- Die Distanz (Metrik) der Route zum Subnetz wird über das Feld *metric count* angegeben.

Um die Routing-Informationen im Netzwerk periodisch weitergeben und ausgefallene Routen zu unerreichbaren Subnetzen erkennen und nach einer gewissen Zeit aus der Routing-Tabelle entfernen zu können, steht der RIP-Router in Abhängigkeit zu drei unterschiedlichen Timern, die in der Standard RIP-Spezifikation [STD53] mit empfohlenen Werten definiert sind:

- Update-Timer: Alle 30 Sekunden sendet der RIP-Router ein periodisches Routing-Update an seine direkt benachbarten RIP-Router. In diesem periodischen Routing-Update ist die Abbildung des kompletten Inhalts seiner Routing-Tabelle enthalten, mit allen Routing-Informationen, die er über die Erreichbarkeit einzelner Subnetze hat. Da Netzwerke, in denen Nachrichten periodisch versendet werden, die Tendenz dazu haben sich zeitlich zu synchronisieren [FloJa94] und so unnötige Belastungsspitzen entstehen, sollte das Intervall des Update-Timers so implementiert sein, dass es zwischen 0 und 5 Sekunden variiert.
- Timeout-Timer: Jeder Eintrag einer Route in der Routing-Tabelle ist mit einem eigenen Timer versehen, der die Dauer seiner Gültigkeit bestimmt. Nach Ablauf des Timeout-Timers gilt die betreffende Route als ausgefallen und das Ziel-Subnetz als Unerreichbar. Der Timeout-Timer wird initialisiert, wenn die Route neu in die Routing-Tabelle eingetragen wird und jedesmal, wenn ihre Gültigkeit über ein eingehendes Routing-Update durch den entsprechenden Nachbarrouter bestätigt wird. Wurde die Route 180 Sekunden lang nicht bestätigt, gilt die Verbindung als ausgefallen und das Netz als unerreichbar. Die Metrik der Route wird dann auf RIP-Infinity (16) gesetzt, womit gleichzeitig der Garbage-Collection-Timer gestartet wird und der Löschprozess beginnt. Solange die Information über die Route in der Routing-Tabelle steht wird sie, egal mit welcher Metrik, auch an die Nachbarrouter weitergegeben.
- Garbage-Collection-Timer: Mit dem Start des Garbage-Collection-Timers beginnt der Löschprozess für eine Route. Mit Ablauf des Garbage-Collection-Timers wird die betroffene Route komplett aus der Routing-Tabelle des RIP-Routers entfernt. Gilt ein Subnetz als unerreichbar und die Route fortan als ausgefallen, dann verbleibt sie noch eine Zeit von 120 Sekunden mit Metrik 16 (RIP-Infinity) in der Routing-Tabelle, um die Information über die Unerreichbarkeit des Subnetzes auch an die benachbarten RIP-Router weiterzugeben, die die Routing-Information in ihre eigene Routing-Tabelle übernommen haben. Trifft ein Routing-Update mit einer alternativen Route zum Subnetz ein, deren Metrik besser ist, bzw. gültig und kleiner RIP-Infinity, so wird der Eintrag der Route in der Routing-Tabelle sofort durch die neue Routing-Information ersetzt, der Garbage-Collection-Timer gestoppt und der Timeout-Timer wieder initialisiert.

Abbildung 2.2 zeigt ein Netzwerk in Form einer Y-Topologie, bestehend aus RIP-Routern. Die weiter unten folgenden Tabellen zeigen den Inhalt, der den einzelnen Routern zugehörigen Routing-Tabellen.

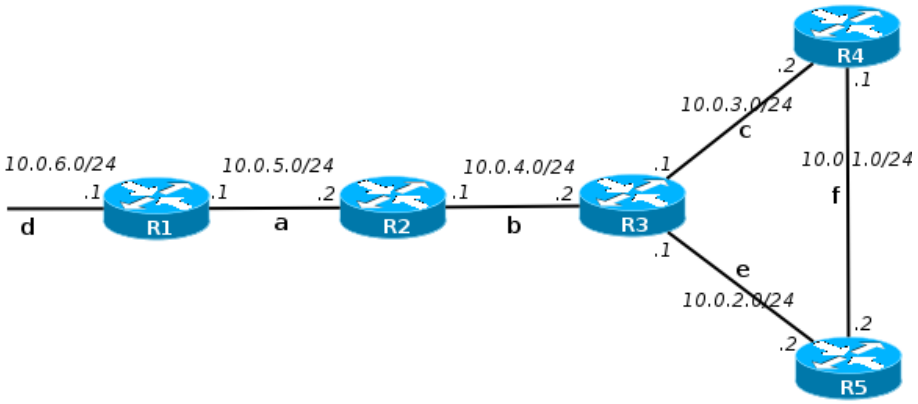


Abbildung 2.2: Y-Topologie eines Netzwerks, bestehend aus RIP-Routern

Die Routing-Tabellen der Router aus der Y-Topologie:

Router R1		
Netz	Metrik	via
d	1	self
a	1	self
b	2	R2
c	3	R2
e	3	R2
f	4	R2

Router R2		
Netz	Metrik	via
d	2	R1
a	1	self
b	1	self
c	2	R3
e	2	R3
f	3	R3

Router R3		
Netz	Metrik	via
d	3	R2
a	2	R2
b	1	self
c	1	self
e	1	self
f	2	R4

Router R4		
Netz	Metrik	via
d	4	R3
a	3	R3
b	2	R3
c	1	self
e	2	R3
f	1	self

Router R5		
Netz	Metrik	via
d	4	R3
a	3	R3
b	2	R3
c	2	R4
e	1	self
f	1	self

Ein Eintrag in der Routing-Tabelle entspricht einer Route und besteht aus dem Ziel-Subnetz, der Metrik zum Ziel-Subnetz und dem ersten Nachbarrouter auf dem Weg zum Ziel-Subnetz. Subnetze die direkt am Router anliegen erhalten eine Metrik von 1 und bekommen das Schlüsselwort „self“ als ersten Nachbarrouter eingetragen. Existiert eine alternative Route so wird immer die Route mit der besten Metrik oder bei gleichgroßen Metriken, die Route die dem Router zuerst über ein Routing-Update angeboten wurde, in die Routing-Tabelle eingetragen. Ein alternative Route zum Subnetz, deren Metrik nicht besser ist als die Metrik der vorhandenen Route, wird unberücksichtigt von RIP verworfen.
In der obigen Topologie bekommt Router R3 über Routing-Updates von R4 und R5 jeweils eine Route zum Subnetz f mit der gleichen Metrik von 2 angeboten. Wird ihm die Route über R4 zuerst bekannt, so übernimmt er R4 als ersten Nachbarrouter für die Route zum Subnetz f. Die gleichwertige Route über R5 verwirft Router R3 dann fortan und berücksichtigt sie nicht mehr, bzw. erst wieder, wenn die Route über R4 ausgefallen ist.

2.2 Routingschleifen und das Counting-to-Infinity Problem

Im folgenden wird das Counting-to-Infinity (CTI) Problem und das damit verbundene Risiko für ein Netzwerk näher erläutert. Das CTI-Problem entsteht aufgrund einer ungünstigen, zeitlichen Sende-Reihenfolge der periodischen Routing-Updates zwischen den RIP-Routern, wobei eine aktuellere Routing-Information, über den Ausfall einer Route und die Unerreichbarkeit des Subnetzes, von einer älteren und nicht mehr gültigen Routing-Information, über die Erreichbarkeit dieses Subnetzes, überschrieben wird. Die aktuellere Information über die Unerreichbarkeit geht dabei verloren. In Abbildung 2.3 ist in vier Schritten dargestellt, wie es zum CTI-Problem über einer Route zu einem Subnetz d kommen kann. Die Topologie entspricht der aus Abbildung 2.2. Die Diagramme rechts daneben zeigen die Auslastung der einzelnen Netze, die ein fortwährend von Router R2 aus gesendeter Ping an eine IP-Adresse aus dem Subnetz d, vor und während das CTI-Problem auftritt, verursacht.

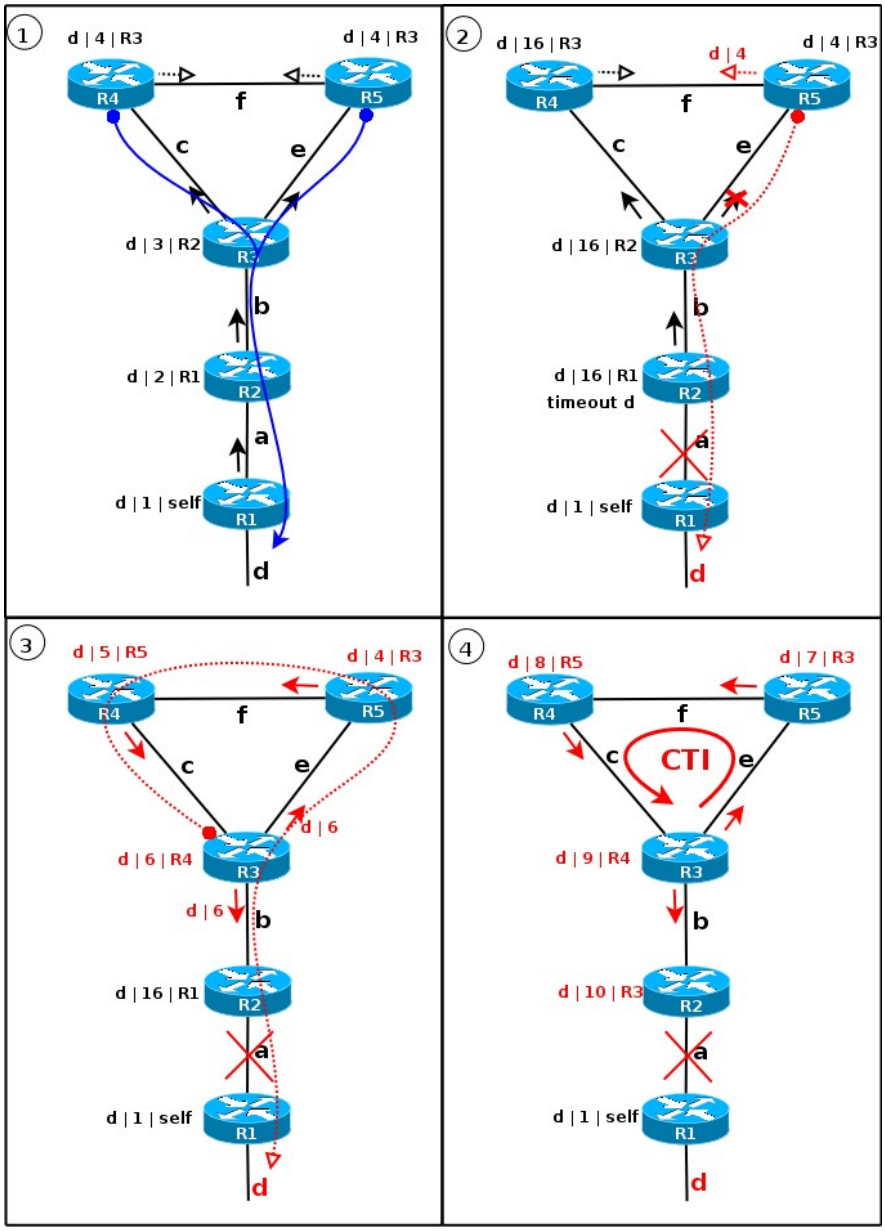
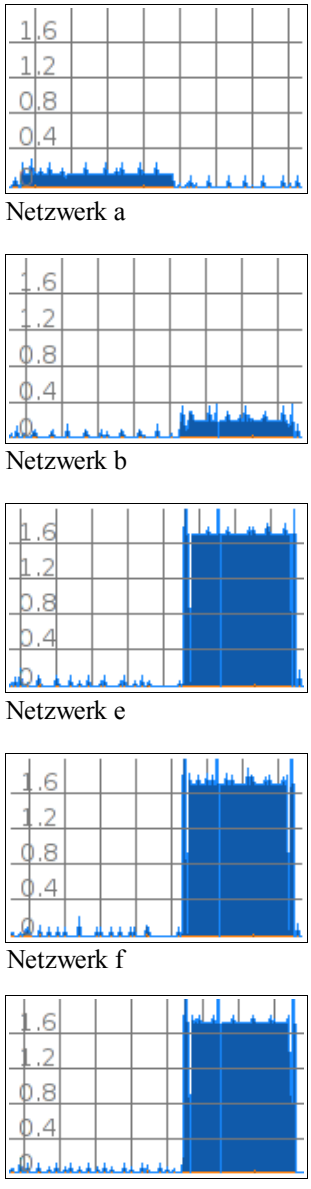


Abbildung 2.3: Das Counting-to-Infinity Problem in der Y-Topologie



Netzwerk a

Netzwerk b

Netzwerk e

Netzwerk f

Netzwerk c

Abbildung 2.4: Auslastung

1) Die Erreichbarkeit des Subnetzes d wird im ganzen Netzwerk verbreitet. Jeder RIP-Router des Netzwerks verfügt über eine gültige Route zum Subnetz d und gibt sie an die RIP-Router der direkten Nachbarschaft weiter. Zu beachten ist, dass sich in der obigen Topologie auch Router R4 und R5 gegenseitig eine Route zum Subnetz d anbieten, das Angebot aber verwerfen, da sie jeweils eine kürzere Route über R3 kennen.

2) Die Verbindung über Subnetz a, zwischen Router R1 und R2, fällt nun aus. Router R1 und das Subnetz d sind nicht mehr von den anderen Routern des Netzwerks zu erreichen. Aufgrund der ausbleibenden Routing-Updates von Router R1 setzt R2 die Metrik seiner Route zum Subnetz d nach Ablauf des Timeout-Timers auf den Wert 16 (RIP-Infinity) und markiert das Subnetz d damit als unerreichbar. Router R3 erhält von R2 die Information über die Unerreichbarkeit des Subnetzes d und sendet dies ebenfalls weiter an seine beiden Nachbarrouter R4 und R5. Während R4 direkt die Information erhält und übernimmt, geht das Routing-Update an R5 unterwegs verloren, was aufgrund der verbindungslosen Kommunikation über UDP von den beteiligten Routern nicht bemerkt werden kann. Router R5 geht somit weiterhin davon aus eine gültige Route zum Subnetz d über Router R3 zu kennen und sendet diese Information auch weiterhin an R4. Dieses Routing-Update enthält nun aber eine ungültige Routing-Information über die Erreichbarkeit des Subnetz d.

3) Router R4 nimmt die Route zum Subnetz d von R5 an, da er sie als gültige Alternative für seine ausgefallene Route sieht. R4 sendet die Routing-Information an R3 weiter, der sie ebenfalls als gültige, alternative Route annimmt. Auch Router R3 gibt die Route zum Subnetz d weiter an seine Nachbarrouter R2 und R5. In dem Moment, in dem R3 die Route zum Subnetz d über R4 in seine Routing-Tabelle eingetragen hat und an seine Nachbarrouter weitersendet, ist eine Routingschleife entstanden.

4) Router R5 nimmt die Route von Router R3 entgegen. R5 ist nach wie vor der Meinung eine gültige Route zum Subnetz d von R3 „gelernt“ zu haben und übernimmt deswegen nun auch die neue Route mit der höheren Metrik von R3. Somit nimmt R5 die Routing-Information entgegen, die von ihm selbst ausgeht. Die Routing-Information zur Erreichbarkeit des Subnetz d wird nun weiter durch die einzelnen Router in der Routingschleife im Kreis gesendet, wobei die zugehörige Metrik bis zum höchstmöglichen Wert, RIP-Infinity (16), hochgezählt wird. Erst wenn dieser Wert erreicht wurde, ist die Unerreichbarkeit des Subnetzes in den beteiligten Routern bekannt und die Routingschleife unterbrochen. Auch die außerhalb der Schleife stehenden Router, wie R2, nehmen die Route zum Subnetz d von einem der Router aus der Netzwerkschleife an. Dies bewirkt letztlich, dass die entstandene Routingschleife wie eine Art Magnet auf sämtlichen Datenverkehr des Netzwerks wirkt, der an eine IP-Adresse aus dem Subnetz d gerichtet ist. Die an der Routingschleife beteiligten Router und Netze könnten somit einer beträchtlichen Mehrbelastung ausgesetzt sein, was schließlich auch einen Ausfall der Netze oder einzelner Netzwerk-Dienste zur Folge haben könnte.

Solange die Metrik der Route zum Subnetz d nicht in mindestens einem der RIP-Router aus der Netzwerkschleife den Wert RIP-Infinity erreicht hat und somit festgestellt wurde, dass das Subnetz nicht mehr erreichbar ist, ist auch eine Routingschleife vorhanden. In dieser Routingschleife wird sämtlicher Datenverkehr, der an eine IP-Adresse aus dem Subnetz d gerichtet ist, zyklisch im Kreis über die drei Router R3, R4 und R5 geleitet, obwohl das Subnetz d längst nicht mehr erreichbar ist. Dabei erhöht sich auch die Datenlast in den einzelnen „Transfernetzen“ (hier: c, e, f) zwischen den drei Routern erheblich. Aufgrund dieser Mehrbelastung ist es gut möglich, dass auch die Routing-Updates selbst während des Sendevorgangs teilweise

oder ganz verloren gehen und so die Routingschleife noch sehr viel länger aufrecht erhalten bleibt und womöglich erst nach Ablauf des Timeout-Timers auf einem der beteiligten RIP-Router unterbrochen wird. Auch ist das CTI-Problem schwer zu lokalisieren, denn es muss nicht in der Nähe des ausgefallenen Netzes auftreten, sondern kann überall in der Topologie, wo eine Gruppe von Routern eine Netzwerkschleife bildet, stattfinden. Die ausgefallene Netzwerk-Komponente, wegen der die Verbindung zum Subnetz d unterbrach, ist aufgrund des CTI-Problems auch schwerer aufzufinden, da die Router im Netzwerk über eine scheinbar gültige Route zum Subnetz d verfügen. Das diese Route Ergebnis einer Routingschleife ist, kann über den einzelnen Router nicht unbedingt erkannt werden.

In Abbildung 2.4 ist die Auslastung der Transfernetze c, e, und f, die dort von einfachen Pings während das CTI-Problem stattfindet verursacht wird, abgebildet. Die Pings werden von Router R2 an eine IP-Adresse aus dem Subnetz d gesendet. Erreichen die Pings ihr Ziel im Subnetz d, so verursachen sie hier im Transfernetz a eine Auslastung von 200 - 300 Bytes/s. Fällt die Route zum Subnetz d aus und kommt es in Folge dessen zum CTI-Problem und somit auch zu einer Routingschleife, so erreichen die Pings in den Transfernetzen c, e und f eine Auslastung von jeweils ca. 1800 bis 1900 Bytes/s. Um welchen Faktor die Auslastung höher ist hängt allerdings letztlich von der Metrik der Routingschleife und von dem eingestellten TTL[5]-Wert der Daten-Pakete ab. In dem obigen Beispiel liegt der TTL-Wert bei 64 was bedeutet das jedes gesendete Ping-Paket jeden Router und jedes Transfernetz in der Schleife ca. 20 mal passiert, bis der TTL-Wert 0 erreicht hat und aufgrund dessen das Ping-Paket verworfen wird.
In den obigen Netzwerklast-Abbildungen ist gut zu erkennen, dass durch eine Routingschleife auch die Netzwerk-Ressourcen sehr stark beeinträchtigt werden können, die am normalen Sendevorgang eigentlich unbeteiligt wären. Weniger das CTI-Problem selbst ist also in diesem Fall das eigentliche Problem, sondern der erheblich erhöhte Datenverkehr aufgrund der Routingschleife und der falschen Erreichbarkeits-Information im Netzwerk.

Im folgenden ist die Ausgabe des Konsolen-Programms „Traceroute“ zu sehen. Traceroute ist eine Diagnose-Software, die den Weg eines Datenpaketes durch das Netzwerk zum Zielhost ermittelt und die IP-Adressen, der auf dem Weg dort hin passierten Netzwerk-Knoten, ausgibt. Auch die von Traceroute zur Ermittlung des Weges gesendeten Pakete verfangen sich in der entstandenen Routingschleife und durchlaufen diese mehrmals bis die eingestellte, maximale Anzahl an Hops einer Strecke, von hier 30, erreicht ist. Die Lokalität der aufgeführten IP-Adressen in der Topologie kann aus Abbildung 2.2 entnommen werden.

```
R2:~# traceroute 10.0.6.1
traceroute to 10.0.6.1 (10.0.6.1), 30 hops max, 38 byte packets
 1  10.0.4.2 (10.0.4.2)  2.451 ms  0.404 ms  0.351 ms
 2  10.0.3.2 (10.0.3.2)  25.027 ms  0.506 ms  0.512 ms
 3  10.0.2.2 (10.0.2.2)  40.169 ms  0.597 ms  0.731 ms
 4  10.0.4.2 (10.0.4.2)  0.568 ms  0.578 ms  0.524 ms
 5  10.0.3.2 (10.0.3.2)  0.677 ms  0.659 ms  0.687 ms
 6  10.0.2.2 (10.0.2.2)  1.238 ms  1.449 ms  0.929 ms
 ...
28  10.0.4.2 (10.0.4.2)  5.545 ms  6.702 ms  7.201 ms
29  10.0.3.2 (10.0.3.2)  2.358 ms  2.264 ms  2.320 ms
30  10.0.2.2 (10.0.2.2)  2.464 ms  2.420 ms  2.951 ms
R2:~#
```

Abbildung 2.5: Ausgabe des Programms Traceroute, während eines CTIs

5 TTL: Time-to-Live: Hop-Counter, der verhindert das unzustellbare Pakete unendlich weiter geroutet werden

2.3 Erweiterungen zur Einschränkung des CTI-Problems

Das Counting-to-Infinity Problem war bereits bei der Spezifikation des RIPv1 Standards [RFC1058] als erhebliche Einschränkung des RIP-Protokolls und Ursache für Instabilitäten in Netzwerken bekannt. Es wurden einige Erweiterungen zu RIP vorgeschlagen die mehr Stabilität und eine schnellere Konvergenz bieten sollten. Diese Erweiterungen können die Gefahr des Auftretens zwar verringern, das CTI-Problem vollständig verhindern können sie allerdings nicht. Im folgenden werden einige dieser Erweiterungen für RIP kurz erläutert.

2.3.1 Triggered-Updates

Bei diesem Vorschlag sollen die RIP-Router unmittelbar nach einer Veränderung in der Routing-Tabelle ein Routing-Update (Response) aussenden, das lediglich die Änderungen enthält. Die Triggered-Updates wurden eingeführt um Veränderungen in der Netzwerk-Topologie schneller unter den RIP-Routern bekannt zu machen. So müssen die Router nicht auf das nächste periodische Routing-Update warten, das immer erst nach Ablauf des Update-Timers folgen würde. Von Nachteil ist allerdings der starke Anstieg der Anzahl an Updates bei Veränderungen im Netzwerk. Um die Anzahl der Routing-Updates trotzdem weitestgehend gering zu halten und nicht für jede Veränderung ein eigenes Update zu senden, warten die RIP-Router eine zufällige Zeitspanne zwischen 0 und 5 Sekunden, bis sie ein Triggered-Update erneut „triggern“. Während das Verzögern des Triggerd-Updates die Information der Routing-Tabelle auf wenige Updates bündelt, erhöht es jedoch gleichzeitig das Risiko, dass ein periodisches Routing-Update eines anderen Routers nicht mehr gültige Informationen, über die Erreichbarkeit eines Subnetzes, zu einem Nachbarrouter sendet und dessen gültige Routing-Information über den Ausfall einer Route überschreibt. Triggered Updates verringern die Dauer des CTI-Problems erheblich, sowie das Risiko des Auftretens, können es jedoch nicht verhindern. Triggered-Updates sollten nach [RFC1812] in RIP-Routern immer aktiviert sein.

2.3.2 Split-Horizon

Mit aktiviertem Split-Horizon sendet ein RIP-Router Informationen über eine Route nicht mehr an den RIP-Router zurück, von dem er sie „gelernt“ hat. Dieser Ansatz verringert nicht nur die Anzahl der zu übermittelnden Routing-Informationen, sondern verhindert auch das Entstehen des CTI-Problems zwischen zwei Routern. Allerdings kann das CTI-Problem nicht auf diese Weise auch in komplexeren Topologien, mit größeren Netzwerkschleifen, verhindert werden. In Topologien in denen mehr als zwei Router eine Netzwerkschleife bilden oder zwei Router über mehr als eine Leitung miteinander verbunden sind, kann es nach wie vor zum CTI-Problem kommen. Nach [RFC1812] sollte Split-Horizon in RIP-Routern immer aktiviert sein.

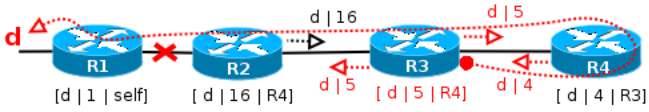


Abbildung 2.5: Mögliche Update-Reihenfolge in einer linearen Topologie ohne Split-Horizon

2.3.3 Split Horizon with poisened reverse

Dies ist eine Erweiterung des Split-Horizon-Mechanismus. Hierbei sendet der RIP-Router die Routing-Information zu dem Router zurück, von dem er sie übernommen hat, allerdings mit der Metrik RIP-Infinity. Dadurch wird garantiert ausgeschlossen, dass zwei Router voneinander eine Route zu einem Subnetz „lernen“ und gegenseitig auf sich zeigen, wodurch sogenannte „Two-Hop-Loops“ entstehen würden. Allerdings erhöht sich mit „poisened reverse“ auch die Anzahl der zu übermittelnden Routing-Informationen wieder und damit auch die Netzwerklast. Split-Horizon allein verhindert im Grunde schon das Entstehen von „Two-Hop-Loops“. Zwar wird in [RFC1812] die Aktivierung von Poison-Reverse empfohlen, ist jedoch im Gegensatz zu Split-Horizon nicht als „Muss“ gekennzeichnet.

2.3.4 Path-Hold-Down-Timer

Der Path-Hold-Down-Timer stellt eine Zeitspanne (von meist 60 Sekunden) nach dem Ausfall einer Route dar, in der ein RIP-Router neue Routing-Informationen über die Erreichbarkeit des Subnetzes ignoriert. Wenn eine Route ausgefallen ist und das Subnetz als unerreichbar gilt, wird eine Hold-Down-Period gewählt, die lange genug ist, bis davon ausgegangen werden kann, dass alle Router im Netzwerk die Unerreichbarkeit des Subnetzes erkannt haben. Erst nach Ablauf dieser Zeitspanne werden alternative Routen zum Subnetz akzeptiert und in die Routing-Tabelle eingetragen. Mit aktiviertem Path-Hold-Down-Timer wird die Existenz von falschen Erreichbarkeiten eines Subnetzes nach dem Ausfall einer Route ignoriert und das CTI-Problem so generell verhindert. Dies ist allerdings auch ein sehr träges Verfahren, da auch gültige Routen während der „Hold-Down-Period“ nicht als alternative Routen akzeptiert und in die Routing-Tabelle eingetragen werden.

2.3.5 RIP-MTI-Algorithmus

Der „RIP with minimal topology information“ (RIP-MTI) Algorithmus ist eine abwärtskompatible Erweiterung zu RIP, die das CTI-Problem vollständig zu beheben verspricht. Der erstmals in [Sch99] vorgestellte RIP-MTI-Algorithmus verändert nicht das bestehende RIP-Nachrichtenformat, sondern bezieht seine benötigten Informationen aus den vorhandenen Routing-Informationen der eingehenden Routing-Updates, die nach der Standard RIP-Spezifikation unberücksichtigt verworfen werden. Der RIP-MTI-Algorithmus kann in einem Netzwerk aus konventionellen RIP-Routern eingesetzt werden und dieses erweitern.
Der RIP-MTI-Algorithmus sammelt Informationen über die am Router anliegenden Netzwerkschleifen der Topologie. Mit Hilfe dieser zusätzlichen Informationen und den vorhandenen Routing-Informationen entscheidet er, ob eine ausgefallene Route aus der Routing-Tabelle mit einer alternativen Route ersetzt werden darf oder ob diese Ersetzung zum CTI-Problem führen kann, weil die alternative Route auf einer ungültigen Information der Erreichbarkeit des Subnetzes basiert. Letztlich erlaubt nur eine Netzwerkschleife die Existenz einer alternativen Route zu einem Subnetz, ermöglicht aber auch das Entstehen von Routing-Schleifen und das Auftreten des CTI-Problems. Zusammen mit dem RIP-MTI-Algorithmus sollte auch Split-Horizon und Triggered-Updates aktiviert sein, um das CTI-Problem verhindern zu können.

2.4 Zusammenfassung

Das Routing Information Protocol (RIP) gilt als der klassische Vertreter der Familie der Distanzvektorprotokolle, da es sich um eine der ersten und einfachsten Umsetzungen des Distanzvektoralgorithmus handelt. Es gehörte einst mit zu den verbreitetsten Routing-Protokollen, wurde aber mittlerweile durch modernere Routing-Protokolle weitestgehend verdrängt. Die Vorteile von RIP haben ihm aber dennoch zu einem Überleben an den Rändern großer Netzwerke und in kleinen, einfachen Netzwerken verholfen. Ein sehr großer Nachteil von RIP ist das Counting-to-Infinity (CTI) Problem, woraus eine erhebliche Einschränkung im Einsatz und in der Erweiterbarkeit der mit RIP gerouteten Netzwerke folgt. Allerdings ist das CTI-Problem für alle Routing-Protokolle, die auf dem Distanzvektoralgorithmus basieren, eine Herausforderung, gegen die mehr oder weniger komplizierte Mechanismen entwickelt wurden. Die Mechanismen die bislang bewiesenermaßen das CTI-Problem verhindern, benötigen eine Veränderung des Nachrichtenformats und sind somit nicht Kompatibel zur bestehenden RIP Spezifikation. Eine RIP-kompatible Lösung des CTI-Problems könnte auch für andere Distanzvektorprotokolle eine umsetzbare Erweiterung sein und diese vereinfachen oder verbessern.
Der RIP-MTI-Algorithmus verspricht eine solche Lösung zu sein, die bislang allerdings noch nicht in produktiven Netzwerksystem eingesetzt wurde und sich somit erst noch im Alltag bewähren muss.

3 RIP with minimal topology information (RIP-MTI)

In diesem Kapitel wird der RIP-MTI-Algorithmus erläutert, den Andreas Schmid in seiner Diplomarbeit [Sch99] zur Vermeidung des Counting-to-Infinity (CTI) Problems vorgeschlagen hat. Die grundlegende Idee dieses Algorithmus ist es, Netzwerkschleifen in der Topologie zu erkennen und dieses Wissen schließlich zu verwenden, um nach Ausfall einer Route gültige und erwünschte alternative Routen von ungültigen Routen zu unterscheiden und letztere abzulehnen, da diese das CTI-Problem verursachen können. Der RIP-MTI-Algorithmus bezieht seine Informationen aus den eintreffenden Routing-Updates. Diese Informationen fliesen als zusätzliche Parameter in den Routing-Prozess mit ein. Das Protokoll selbst, also Struktur und Ablauf der Kommunikation zwischen den RIP-Routern, wird nicht verändert.

3.1 Definition der Grundlagen

Im folgenden werden einige Definitionen vorgestellt, die im weiteren Verlauf der Ausarbeitung immer wieder Verwendung finden.

- Definition des Netzwerk-Models:
  Netzwerke bestehen aus Routern und Subnetzen, die über Interfaces miteinander verbunden sind. Ein Netzwerk kann durch einen Graphen dargestellt werden, wobei die Knoten des Graphen die Router darstellen und die Kanten die Subnetze. Ein Interface ist die Verbindung zwischen einem Knoten und einer Kante.
  Ein Netzwerk G(RO, SN, IF) besteht somit aus der Menge der Router RO=$\{r_1, r_2, ..., r_n\}$, der Menge der Subnetze SN=$\{s_1, s_2, ..., s_n\}$ und der Menge der Interfaces IF=$\{if_1, if_2, ..., if_n\}$. Ein Router $r \in RO$ ist durch die Menge seiner Interfaces definiert, so dass gilt:
  $r = \{A, B, C, ... \}$, $r \subseteq IF$ und $\forall A \in IF, A \in r_i \Rightarrow A \notin r_j$ für $r_i, r_j \in RO, i \neq j$. Ein Interface ist mit genau einem Router verbunden. Die Interfaces sind mit den Subnetzen verbunden und somit ist die Topologie des Netzwerkgraphen über die Relation $CON \subseteq IF \times SN$ definiert, wobei gilt $(if_i, s_j) \in CON$ mit $if_i \in IF$ und $s_j \in SN$, genau dann wenn das Interface if_i mit dem Subnetz s_j verbunden ist.

Der RIP-MTI-Algorithmus basiert auf dem Ansatz, spezielle Pfade zwischen den Routern und Subnetzen in der Netzwerk-Topologie erkennen und voneinander unterscheiden zu können.

Ein Pfad ist wie folgt definiert.

- Definition Pfad:
  Ein Pfad $P^{i,j}$ ist der Weg zwischen einem Router i und einem Router j und kann als Sequenz von Hops beschrieben werden. Ein Hop ist in diesem Sinne als Triple $H^{i,k} = (O, s, I)$ definiert, mit einem Ausgangs-Interface O, einem Subnetz s und einem Eingangs-Interface I, wobei gilt: (O, s), (I, s) $\in$ CON.

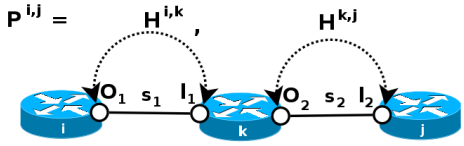


Abbildung 3.1: Ein Pfad besteht aus Hops

  Ein Hop beschreibt einen Pfad oder den Abschnitt eines Pfades der bei Router $i \in RO$ beginnt, diesen über das Interface $O \in i$ verlässt, das am Router anliegende Subnetz $s \in SN$ passiert und über das Interface $I \in k$ den benachbarten Router k betritt.

Somit ist ein Pfad $P^{i,j}$ der bei Router i, über dessen Interface A, beginnt und bei Router j, über dessen Interface B endet, über die Hops zwischen diesen beiden Routern beschrieben.

$$P^{i,j}_{A,B}=(H_1^{i,i+1}, H_2^{i+1,i+2}, \ldots, H_L^{j-1,j})=((O_1=A, s_1, I_1),(O_2, s_2, I_2,), \ldots,(O_L, s_L, I_L=B))$$

Die Metrik des Pfades $P^{i,j}_{A,B}$ ist die Distanz zwischen den Routern i und j mit der Länge $m^{i,j}_{A,B}=L$, was der Anzahl der Hops in der Pfad-Beschreibung entspricht.

Eine Route ist ein spezieller Pfad, der nicht an einem Interface eines Routers endet, sondern diesen durchläuft und erst in einem direkt am Router anliegenden Subnetz endet.

- Definition Route:
  Eine Route ist ein spezieller Pfad der in einem Subnetz endet. Die Route ist, wie der Pfad, als eine Sequenz von Hops definiert, wobei der letzte Hop keine Interface Gegenstelle besitzt, was mit einem „*“ gekennzeichnet wird. Ein Hop, ausgehend von einem Router j in das Subnetz d, kann mit $H^{j,d}$ =(O, d, *) notiert werden. Eine Route, die bei einem Router i beginnt, über Interface A diesen verlässt und bei einem Subnetz d, welches hinter einem Router j liegt, endet, wird über die dazwischenliegenden Hops beschrieben.

  $$P^{i,d}_{A}=(H_1^{i,i+1}, H_2^{i+1,i+2}, \ldots, H_L^{j,d})=((O_1=A, s_1, I_1),(O_2, s_2, I_2,), \ldots,(O_L, d, *))$$

  Die Metrik einer Route $P^{i,d}_{A}$ ist die Distanz zwischen dem Router i und dem unmittelbar vor dem Ziel-Subnetz d anliegenden Router j und hat die Länge $m^{i,d}_{A}=L$, was der Anzahl der Hops in der Pfad-Beschreibung entspricht. Routen mit kleineren Metriken sind besser als solche mit größeren Metriken und werden vom Routing-Prozess bevorzugt.

Für den RIP-MTI-Algorithmus sind vor allem solche Routen interessant, die sich einem Router als alternative Routen zu einem Subnetz anbieten und sich durch das Ausgangs-Interface des Routers voneinander unterscheiden. Werden zwei solcher alternativer Routen, die beide von einem Router i aus zu einem gleichen Subnetz d führen und sich dort letztlich wieder treffen, miteinander kombiniert, so entsteht ein Pfad der bei einem Interface A des Routers i beginnt, das Subnetz d passiert und bei einem Interface B des Routers i wieder endet, wobei die Interfaces A und B des Routers i verschieden sind. Ein solcher Pfad beschreibt eine an den Router i anliegende Netzwerkschleife über dem Subnetz d.

- Definition Routen-Kombination:
  Eine Routen-Kombination $P^{i,d,i}_{A,B}$ ist die Kombination zweier Routen $P^{i,d}_{A}$ und $P^{i,d}_{B}$ des Routers i über einem Subnetz d. Das Subnetz d ist von Router i aus mit jeweils einer Route über Interface A sowie über Interface B zu erreichen, wobei gilt A ≠ B, A, B ∈ IF. Der Pfad einer Routen-Kombination, die bei einem Interface A des Routers i beginnt, das Subnetz d passiert und bei einem Interface B des Routers i wieder endet, wird wie folgt beschrieben.

  $$P^{i,d,i}_{A,B}=((O_1=A, s_1, I_1),(O_2, s_2, I_2), \ldots,(O_k, d, I_{k+1}), \ldots,(O_L, s_L, I_L=B))$$

  Die Metrik $m^{i,d,i}_{A,B}$ der Routen-Kombination $P^{i,d,i}_{A,B}$ berechnet sich aus der Anzahl der Hops in der Beschreibung des Pfades. Sie kann aus der Metrik $m^{i,d}_{A}$ der Route über Interface A und der Metrik $m^{i,d}_{B}$ der Route über Interface B berechnet werden: $m^{i,d,i}_{A,B}=m^{i,d}_{A}+m^{i,d}_{B}-1=L$.
  Der Abschnitt des Pfades, der den Hop am Subnetz d beschreibt und beide Routen $P^{i,d}_{A}$ und $P^{i,d}_{B}$ abschließt, wird bei der Kombination der beiden Routen zu einem einzigen Hop zusammengefasst und muss entsprechend von der Metrik der Routen-Kombination abgezogen werden.

In Abbildung 3.2 (1) ist eine Topologie dargestellt, in der das Subnetz d in einer Netzwerkschleife liegt und für Router i aus zwei verschiedenen Richtungen zu erreichen ist. Wenn die Route über Interface B zum Subnetz d ausfällt, so existiert für den Router i, aufgrund der vorhandenen Netzwerkschleife, eine alternative Route über Interface A, die er direkt im Anschluss, nach dem Ausfall der bisher verwendeten Route über Interface B, verwenden kann.

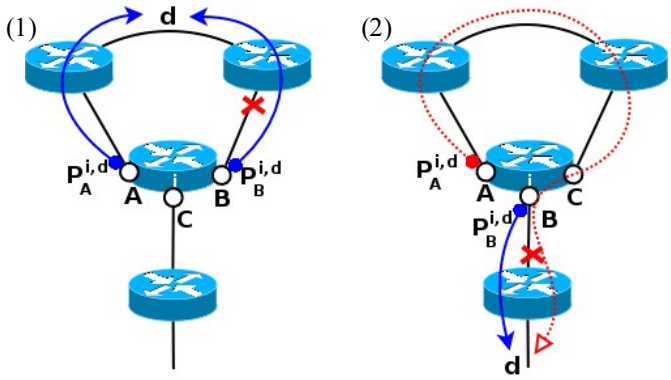


Abbildung 3.2: Gültige und ungültige Ersatzroute für Subnetz d

In Abbildung 3.2 (2) liegt Subnetz d nun an einer anderen Stelle in der Topologie. Fällt hier in Router i die Route zum Subnetz d aus, so darf es keine alternative Route geben, da Subnetz d für Router i nicht aus zwei verschiedenen Richtungen erreichbar ist. Allerdings können sich unter ungünstigen Umständen alte, ungültige Informationen über die Erreichbarkeit des Subnetzes gegenüber der neuen Information der Unerreichbarkeit im Netzwerk durchsetzen. Die beteiligten RIP-Router können in diesem Fall nicht unterscheiden, welche Information aktueller ist, da sich aus den eintreffenden Routing-Updates lediglich der nächste Nachbarrouter und die Metrik der Route zum Subnetz d auslesen lassen. Daher ist aus Sicht des Routers i die Situation in Abbildung 3.2 (2) die gleiche wie in (1): Die bisherige Route zu einem Subnetz d über das Interface B geht verloren und eine alternative Route zum Subnetz wird über das Interface A angeboten. Nimmt der Router i aus Abbildung 3.2 (2) allerdings die vermeintliche alternative Route über Interface A an und trägt sie in sein Routing-Tabelle ein, so kann es hier zum CTI-Problem kommen.

Der RIP-MTI-Algorithmus kann anhand der Metrik $m_{A,B}^{i,d,i}$ der Routen-Kombination $P_{A,B}^{i,d,i}$ erkennen, ob die angebotene alternative Route $P_A^{i,d}$ mit ihrer Metrik $m_A^{i,d}$ überhaupt existieren dürfte. Dafür unterschiedet der RIP-MTI-Algorithmus zwischen unterschiedlichen Routen-Kombinationen, die im folgenden erläutert werden.

- Definition Simple-Loop:
  Ein Simple-Loop entsteht aus der Kombination zweier Routen, die an zwei verschiedenen Interfaces A und B des Routers i beginnen und beide direkt zum Subnetz d führen, ohne den Router i selbst zu durchqueren.
  Ein Simple-Loop ist eine Routen-Kombination $P_{A,B}^{i,d,i}$, die wie folgt definiert ist: A, B ∈ i, O_l=A, I_l =B, ∃ n: 1≤ n≤L, s_n = d und ∀ I_j: 1≤ j< L, I_j ∉ i.
  Die Menge der Simple-Loops wird kurz mit SIL bezeichnet. Ist eine Routen-Kombination ein Simple-Loop, so gilt $P_{A,B}^{i,d,i}$ ∈ SIL. Die Metrik des Simple-Loops entspricht der Metrik der Routen-Kombination: $silm_{A,B}^{i,d,i} = m_{A,B}^{i,d,i} = L$.

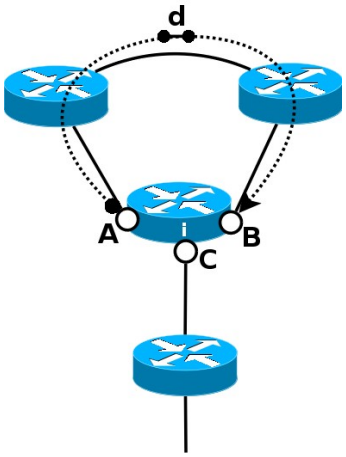


Abbildung 3.3: Simple-Loop

Der RIP-MTI-Algorithmus ist nicht an allen erkennbaren Simple-Loops zwischen zwei Interfaces interessiert, sondern nur an den jeweils kleinsten mit der besten Metrik. Diese sind auf zwei verschiedene lokale Parameter verteilt, die vom RIP-MTI-Algorithmus unterschiedlich verwendet werden.

- Minimal-Simple-Loop-Metric (MSILM):
  Die Minimal-Simple-Loop-Metric $msilm^{i}_{A,B}$ bezeichnet die Metrik des „kleinsten“ Simple-Loops $P^{i,d,i}_{A,B} \in$ SIL zwischen zwei Interfaces A und B.
  Es gilt: $msilm^{i}_{A,B} = min\{ silm^{i,d,i}_{A,B} \text{ für alle Subnetze } d \}$
  Die Definition ist symmetrisch: $msilm^{i}_{A,B} = msilm^{i}_{B,A}$. Der MSILM-Wert wird mit RIP-MTI-Infinity initialisiert, was der Kombination zweier ausgefallenen Routen mit der Metrik RIP-Infinity entspricht, also 2 * RIP-Infinity – 1. Bei einer RIP-Infinity Metrik von 16, wie von der RIP-Spezifikation vorgesehen, beträgt RIP-MTI-Infinity somit 31. Wurde also kein Simple-Loop zwischen zwei Interfaces A und B erkannt, beträgt der MSILM-Wert hier 31.
- Minimal-Return-Path-Metric (MRPM):
  Die Minimal-Return-Path-Metrik $mrpm^{i}_{A}$ bezeichnet die Metrik des „kleinsten“ Simple-Loops $P^{i,d,i}_{A,B} \in$ SIL zwischen einem Interface A des Routers i und allen anderen Interfaces des Routers. Der MRPM-Wert ist vom MSILM-Wert abgeleitet. Während der MSILM-Parameter die Metrik des „kleinsten“ Simple-Loops zwischen zwei spezifischen Interfaces angibt, handelt es sich beim MRPM-Wert um den „kleinsten“ Simple-Loop eines Interfaces: $mrpm^{i}_{A} = min\{ msilm^{i}_{A,X} \text{ für alle Interfaces } X \neq A \text{ des Routers } i \}$
  Der MRPM-Wert eines Interfaces A ist das Minimum über $msilm^{i}_{A,X}$ mit allen Interfaces X des Routers i. Der MRPM-Wert wird mit dem Wert RIP-MTI-Infinity (31) initialisiert. Somit ist hinter einem Interface mit dem MRPM-Wert RIP-MTI-Infinty entweder kein Simple-Loop vorhanden oder dieser wurde noch nicht erkannt.

Die erkannten MSILM- und MRPM-Werte werden jeweils in einer eigenen lokalen Tabelle, im weiteren MSILM- und MRPM-Tabelle genannt, im RIP-MTI-Router gespeichert.
Bei Ausfall einer Route kann der Router nur über die Interfaces eine gültige alternative Route zum Subnetz erwarten, die mit dem Ausgangs-Interface der zuvor ausgefallenen Route über einen Simple-Loop in Verbindung stehen. Trifft eine alternative Route über ein Interface, ohne Simple-Loop Verbindung zum Interface der ausgefallenen Route, ein, so handelt es sich mit sehr hoher Wahrscheinlichkeit um eine ungültige Route, deren Annahme zum CTI-Problem führen wird. Diese ungültige Route zum Subnetz basiert auf der zuvor ausgefallenen Route, die der Router selbst an seine benachbarten Router in der Netzwerkschleife weitergegeben hat und die dort aufgrund einer zeitlich ungünstigen Sende-Reihenfolge der Routing-Updates während des Ausfallvorgangs entstanden ist. Diese ungültige Route basiert somit auf einer alten Erreichbarkeitsinformation des Routers selbst. Auch diese ungültige, alternative Route und die zuvor ausgefallene Route können zu einer speziellen Routen-Kombination zusammengefasst werden.

- Definition Source-Loop:
  Ein Source-Loop ist die Kombination zweier Routen zum Subnetz d, wobei eine der beiden Routen auf einer ungültigen Erreichbarkeitsinformation basiert und den Router i zuvor bereits schon einmal passierte. Ein Source-Loop ist eine Routen-Kombination $P^{i,d,i}_{A,B}$, die wie folgt definiert ist:
  $A, B \in i, O_l = A, I_l = B, \exists\, n: 1 \leq n \leq L, s_n = d$ und $\exists\, I_j: 1 \leq j < L, I_j \in i$.
  Die Menge der Source-Loops wird SOL bezeichnet. Ist eine Routen-Kombination ein Source-Loop, so gilt $P^{i,d,i}_{A,B} \in$ SOL. Die Metrik des Source-Loop entspricht der der Routen-Kombination: $solm^{i,d,i}_{A,B} = m^{i,d,i}_{A,B} = L$.

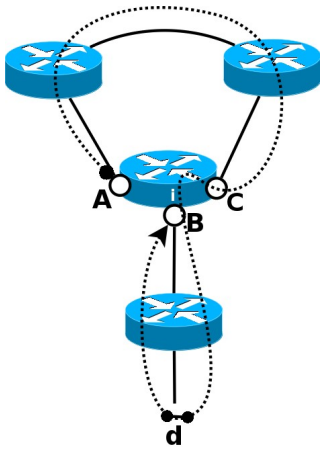


Abbildung 3.4: Source-Loop

Ein Source-Router ist in diesem Zusammenhang der Router in der Netzwerkschleife, der dem ausgefallenen Subnetz am nächsten ist, bzw. der den anderen Routern der Netzwerkschnleife die Route zum Subnetz übermittelt hat. Der Source-Router befindet sich beim Konzept des RIP-MTI-Algorithmus an der Schlüsselposition im Netzwerk. Von seinem Verhalten hängt es ab, ob das Entstehen des CTI-Problems nach Ausfall einer Route verhindert werden kann.

Der RIP-MTI-Algorithmus unterscheidet zwischen zwei speziellen Source-Loops, die abhängig von der vorhandenen Topologie und der Reihenfolge der Routing-Updates zwischen den Routern entstehen können.

- Definition Y-Kombination:
  Die Y-Kombination ist ein spezieller Source-Loop (3), über einem Subnetz d, der aus den beiden Routen-Kombinationen $P_{A,C}^{i,e,i}$ und $P_{B,B}^{i,d,i}$ besteht, wobei gilt: e, d ∈ SN, e ≠ d und A, B, C ∈ IF, A ≠ B mit i={A, B, C}. Die Y-Kombination entsteht letztlich aus der Kombination der beiden Routen zum Ziel-Subnetz d $P_A^{i,d}$ über Interface A und $P_B^{i,d}$ über Interface B, wobei die Route $P_A^{i,d}$ allerdings eine direkte Fortführung der Route $P_B^{i,d}$ durch den Router i ist. Die Metrik $m_A^{i,d}$ basiert somit auf der Metrik $m_B^{i,d}$ und es gilt $m_A^{i,d} \geq m_B^{i,d} + c$ mit $c \geq m_{A,C}^{i,e,i}$. Durch die Aufnahme der Route $P_A^{i,d}$ in die Routing-Tabelle des Routers i

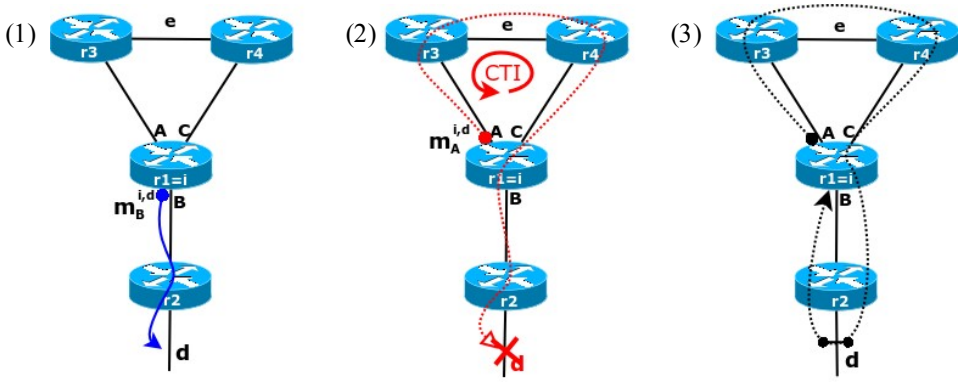


Abbildung 3.5: Y-Kombination

Abbildung 3.5 zeigt den Ablauf des Entstehens einer Y-Kombination. Router i erhält eine Route zum Subnetz d über Interface B mit der Metrik $m_B^{i,d}$ von Router r2 und gibt diese Route an seine beiden benachbarten Router r3 und r4 weiter (1). Die Route zum Subnetz d fällt nun aus und wird von Router i mit Metrik RIP-Infinity (16) weitergegeben. Router r3 übernimmt die Information über den Ausfall der Route von r1, aber noch bevor auch r4 die Information übernimmt, sendet r4 seine Route mit mittlerweile veralteter und ungültiger Metrik an r3, der diese Route als bessere gegenüber seiner, mit Metrik RIP-Infinity, übernimmt und wiederum an Router i weiter sendet (2). Nach der Standard RIP-Spezifikation würde Router i hinter der Metrik $m_A^{i,d}$ eine gültige, alternative Route von r3 vermuten und annehmen.

- Definition X-Kombination:
  Die X-Kombination ist ein spezieller Source-Loop (4), über einem Subnetz d, der aus zwei miteinander verbundenen Simple-Loops $P_{A,C}^{i,e,i}$ und $P_{B,D}^{i,d,i}$ besteht, wobei gilt e, d ∈ SN, e ≠ d und A, B, C, D ∈ IF, A ≠ B, A ≠ C, C ≠ D mit i={A, B, C, D}. Die X-Kombination entsteht aus der Kombination der beiden Routen zum Subnetz d, $P_A^{i,d}$ über Interface A und

$P_B^{i,d}$ über Interface B, wobei allerdings die Route $P_A^{i,d}$ eine Weiterführung der Route $P_D^{i,d}$ über Interface D des Routers i ist. Die Route $P_D^{i,d}$ ist jedoch für den Router i nicht mehr verfügbar und wurde zuvor von der Route $P_B^{i,d}$ verdrängt. Es gilt $m_A^{i,d} \geq m_D^{i,d} + c$ mit $c \geq m_{A,C}^{i,e,i}$ und $m_B^{i,d} > m_D^{i,d}$. Im Gegensatz zur Y-Kombination existieren zwischen den beiden Metriken $m_A^{i,d}$ und $m_B^{i,d}$ keinerlei Zusammenhänge. Durch die Annahme der Route $P_A^{i,d}$ kann das CTI-Problem entstehen.

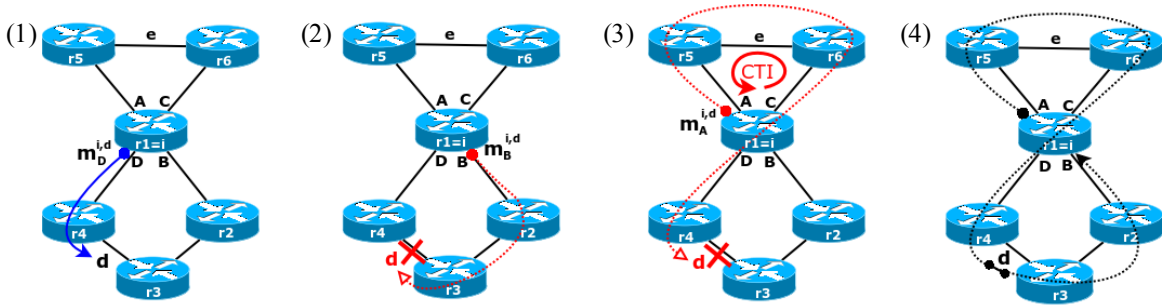


Abbildung 3.6: X-Kombination

Abbildung 3.6 zeigt den Ablauf des Entstehens einer X-Kombination. Router i erhält eine Route zum Subnetz d über Interface D mit der Metrik $m_D^{i,d}$ von Router r4 und gibt diese Route an seine beiden benachbarten Router r5 und r6 weiter (1). Das Subnetz d fällt nun aus und die Route wird von Router i mit der Metrik RIP-Infinity weitergegeben. Router r5 übernimmt die neue Routing-Information von r1, aber noch bevor auch r6 die neue Information erreicht, sendet r6 seine Route mit mittlerweile ungültiger Metrik an Router r5, der sie als bessere gegenüber seiner mit Metrik RIP-Infinity übernimmt. In der gleichen Zeit übernimmt Router i von r2 eine vermeintlich alternative, aber tatsächlich falsche, Route zum nicht mehr erreichbaren Subnetz d über Interface B mit Metrik $m_B^{i,d}$ (2). Kurz darauf registriert jedoch Router r2 den Ausfall der Route zum Subnetz d und überschreibt sie bei Router i wieder mit Metrik RIP-Infinity. Router r5 sendet nun seine Route an Router i weiter (3). Im Gegensatz zur Y-Kombination basiert diese Route nun nicht mehr auf der letzten gültigen Erreichbarkeitsinformation von Router i und kann daher auch nicht genauso behandelt werden. Nach der Standard RIP-Spezifikation würde Router i die Route $P_A^{i,d}$ mit der Metrik $m_A^{i,d}$ von r5 annehmen, was zum CTI-Problem führen könnte.

Im Zusammenhang mit dem Begriff Source-Loop wird nun auch der Begriff Routing-Loop vorgestellt.

- Definition Routing-Loop:
  Ein Routing-Loop ist die Voraussetzung für das Auftreten des Counting-to-Infinity Problems. Ein Routing-Loop beschreibt eine spezielle Route mit der Metrik $m_A^{i,d}$ (bzw. eine Routen-Kombination mit der Metrik $m_{A,B}^{i,d,i}$) die einen beliebigen Router mindestens zweimal durchläuft.

Ziel des RIP-MTI-Algorithmus ist es, solche Routen-Kombinationen die einen Source-Loop beschreiben von solchen Kombinationen die einen Simple-Loop beschreiben zu unterscheiden, um Routing-Loops und somit letztlich das Counting-to-Infinity Problem zu verhindern.

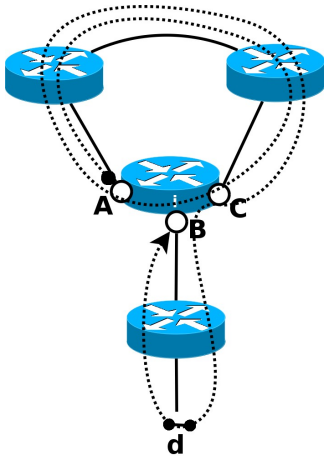


Abbildung 3.7: Routing-Loop

3.2 Simple-Loops erkennen und Source-Loops verhindern

Ein Source-Loop ist eine Routen-Kombination über einem Subnetz d, die einen Router i bei einem Interface B verlässt, ihn anschließend wieder über ein beliebiges Interface betritt, ihn ein weiteres mal über ein beliebiges Interface verlässt, um den Router schließlich über das Interface A wieder zu erreichen. Nach dieser Definition des Ablaufs beginnt und endet ein Source-Loop immer an den beiden Interfaces A und B und durchläuft den Router über höchstens zwei weitere beliebige Interfaces (any ∈ IF). In Abbildung 3.8 ist eine einfache Möglichkeit der Herleitung aller Source-Loops abgebildet. Durch entsprechende Besetzung der any-Interfaces auf die anderen Interfaces des Routers können alle möglichen Source-Loops abgeleitet werden.
Es gibt lediglich neun verschiedene Möglichkeiten für eine Routen-Kombinationen einen Source-Loop zu beschreiben. In Abbildung 3.9 sind diese neun Möglichkeiten für einen Source-Loop aufgelistet, zusammen mit der Technik, durch die ihr auftreten vermieden werden kann. Es existiert kein weiterer Source-Loop der nicht zu einem der neun abgebildeten analog wäre.

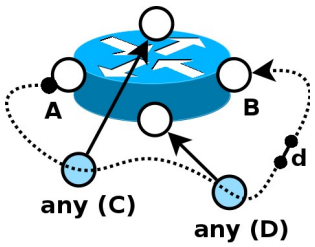


Abbildung 3.8: Herleitung der Source-Loops

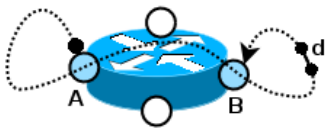
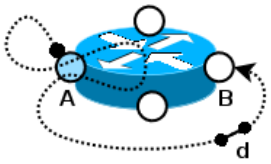
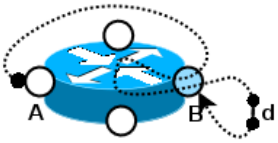
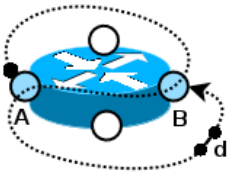
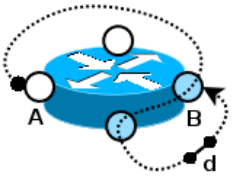
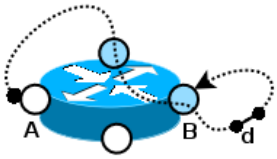
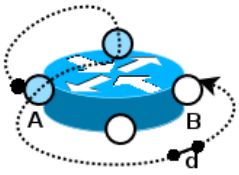
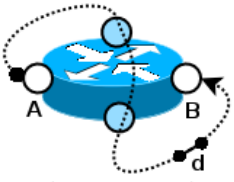
Source-Loop Pfade	A,*,B,B	A,*,A,B	A,*,any,B
A,A,*,B	1) Y-Test + ESH  (A, A, B, B)	2) ISH + ESH  (A, A, A, B)	3) ESH  (A, A, any, B)
A,B,*,B	4) Y-Test + ISH  (A, B, B, B)	5) X-Test  (A, B, A, B)	6) X-Test  (A, B, any, B)
A,any,*,B	7) Y-Test  (A, any, B, B)	8) X-Test  (A, any, A, B)	9) X-Test  (A, any, any, B)

Abbildung 3.9: Auflistung aller möglichen Source-Loops

Ziel des RIP-MTI-Algorithmus ist es nun, all diese Source-Loops zu erkennen und die Aufnahme der zugehörigen Route zu verhindern, um somit auch das Counting-to-Infinity Problem zu vermeiden. Einige der hier aufgeführten Source-Loops werden bereits durch die Aktivierung des Split-Horizon-Mechanismus in den RIP-Routern verhindert. Split-Horizon ist Bestandteil der Spezifikation des RIP Standards und besagt, dass ein Router eine Route nicht mehr an den Router zurücksenden darf, von dem er die Route übernommen hat (Kapitel 2.3.2). In [RFC1812] wird die generelle Aktivierung von Split-Horizon in RIP-Routern empfohlen.

Bei der Vermeidung eines Source-Loops wird zwischen zwei Wirkungsweisen des Split-Horizon unterschieden, abhängig davon in welchem Router Split-Horizon bzgl. der Route wirkt.

- External Split Horizon (ESH):
  Der Source-Loop wird aufgrund des aktivierten Split-Horizon des benachbarten RIP-Routers verhindert, der nicht zulässt, dass die Route wieder zu dem Source-Router i zurückgesendet wird, da er sie von diesem übernommen hat. In Abbildung 3.9 werden so die Source-Loops, die mit ESH gekennzeichnet sind, vermieden.

In Abbildung 3.10 ist eine Topologie mit dem Source-Loop (A, A, any, B) aus Abbildung 3.9 (3) zu sehen. Router r2 hat die Route zum Subnetz d von Router i übernommen. Fällt nun die Route zum Subnetz d aus, dann wird Router r2 aufgrund seines aktivierten Split-Horizon kein Routing-Update dieser Route an Router i senden. Ein Source-Loop über der ausgefallenen Route zum Subnetz d wird in diesem Fall durch Router r2 verhindert.

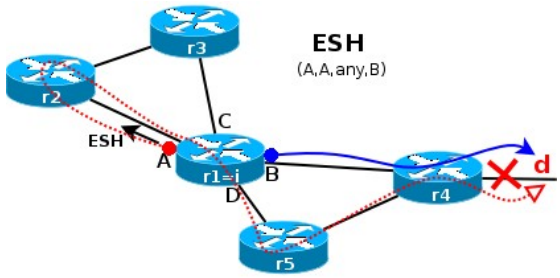


Abbildung 3.10: ESH in r2 verhindert den Source-Loop

- Internal Split Horizon (ISH)
  Hier wird der Source-Loop durch den Router i selbst verhindert, der die Route zum Subnetz d nicht wieder zu dem Router zurücksendet, von dem er sie übernommen hat. In Abbildung 3.9 sind somit die Source-Loops, die mit ISH gekennzeichnet sind ausgeschlossen.

In Abbildung 3.11 ist eine Topologie mit dem Source-Loop (A,B,B,B) aus Abbildung 3.9 (4) zu sehen. Fällt die Route zum Subnetz d aus, dann wird Router i kein falsches Routing-Update an den benachbarten Router r4 senden, von dem er die Route übernommen hat. Der Source-Loop wird hier durch den Router i verhindert.

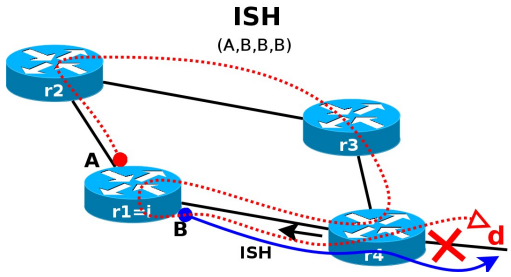


Abbildung 3.11: ISH in r1 verhindert den Source Loop

Die Aktivierung des Split-Horizon in den RIP-Routern verhindert vier der neun möglichen Source-Loops. Es werden allerdings nur solche Source-Loops $P_{A,B}^{i,d,i} \in SOL$ verhindert, von denen die Route $P_A^{i,d}$ ein und denselben Hop zweimal direkt hintereinander durchqueren möchte, bzw. ein Interface des Routers i der Route $P_A^{i,d}$ zweimal hintereinander, in unterschiedlicher Richtung, durchlaufen werden soll.

Die verbliebenen fünf Source-Loops können als X- und Y-Kombinationen identifiziert werden. A. Schmid hat in [Sch99] zwei Gleichungen vorgestellt mit deren Hilfe diese speziellen Typen von Source-Loops aufgrund der Metriken der beiden beteiligten Routen $P_A^{i,d}$ und $P_B^{i,d}$ erkannt und vermieden werden können. Dafür müssen zuvor die in der näheren Umgebung des Routers vorhandenen Simple-Loops erkannt worden sein, da die beiden Gleichungen auf dem MRPM-Parameter aufbauen.

Die Gleichungen werden hier X- und Y-Test genannt und stellen die Kernfunktion des RIP-MTI-Algorithmus bei der Vermeidung des CTI-Problems dar.

- Y-Test: Der Y-Test prüft eine Routen-Kombination $P_{A,B}^{i,d,i}$ mit der Metrik $m_{A,B}^{i,d,i}$ darauf, ob sie eine Y-Kombination ist. Sie ist keine Y-Kombination, wenn die folgende Gleichung erfüllt ist:

$$mrpm_A^i + m_B^{i,d} > m_A^{i,d} \qquad (1)$$

  Die Summe aus der MRPM-Metrik des Interface A $mrpm_A^i$ und der Metrik der Route über Interface B $m_B^{i,d}$ geben die Metrik der kürzest möglichen Y-Kombination über den beiden Interfaces A und B vor. Ist die Metrik der neuen Route über Interface A $m_A^{i,d}$ kleiner, so kann keine Y-Kombination vorliegen.
  Gilt also die Gleichung in (1), dann ist die Routen-Kombination $P_{A,B}^{i,d,i}$ mit der Metrik $m_{A,B}^{i,d,i}$ zu kurz um eine Y-Kombination zu sein und wäre somit ein erwünschter Simple-Loop.

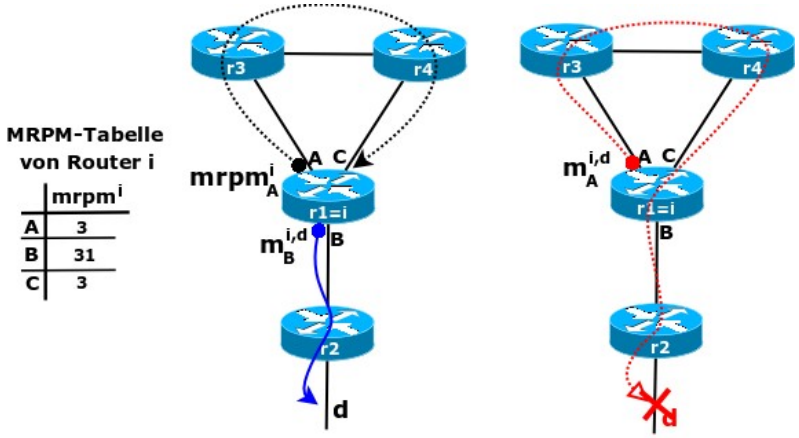


Abbildung 3.12: Der Y-Test erkennt die Y-Kombination

Für das Beispiel aus Abbildung 3.12 gelten die Metriken: $3+2>5\,(f)$.
Somit wird hier kein Simple-Loop erkannt und eine alternative Route $P_A^{i,d}$ mit der Metrik $m_A^{i,d}=5$ würde abgelehnt werden.

Der Y-Test benötigt die MRPM-Metrik des Interfaces, um zu überprüfen, ob eine Routen-Kombination ein Simple-Loop ist oder nicht. Die MRPM-Metrik selbst wiederum ist die Metrik des kleinsten Simple-Loops eines Interfaces. Somit benötigt der Y-Test einen Simple-Loop um eine Routen-Kombination als Simple-Loop zu erkennen oder als Y-Kombination zu verwerfen, was die Frage nach der Initialisierung des Y-Tests aufwirft. Hier wird stets davon ausgegangen, dass es in jeder Netzwerkschleife mindestens ein Subnetz d gibt, welches einem Router i in der Topologie genau gegenüberliegt und so immer zwei verschiedene Routen zum Subnetz über die Interfaces A und B des Routers i existieren, deren Metriken entweder die gleiche Länge haben oder maximal um einen Hop unterschiedlich sind. Mit Hilfe des Y-Tests kann somit der initiale Simple-Loop, der am Router i anliegt, gefunden werden. Für die Initialisierung des Y-Tests gilt somit:

$$2+m_B^{i,d} > m_A^{i,d} \;\rightarrow\; 2>\left|m_A^{i,d}-m_B^{i,d}\right|$$

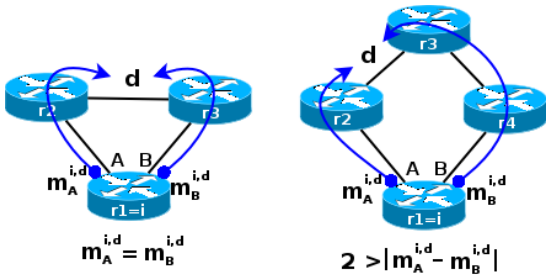


Abbildung 3.13: Finden des initialen Simple-Loop

• X-Test: Eine Routen-Kombination $P_{A,B}^{i,d,i}$ mit der Metrik $m_{A,B}^{i,d,i}$ ist keine X-Kombination, wenn die folgende Gleichung erfüllt ist:

$$mrpm_A^i + mrpm_B^i > m_A^{i,d} + m_B^{i,d} - 1 \qquad (2)$$

Die Summe der MRPM-Werte der Interfaces A $mrpm_A^i$ und B $mrpm_B^i$ geben die Metrik der kürzest möglichen X-Kombination über diesen beiden Interfaces vor. Ist die Routen-Kombination $P_{A,B}^{i,d,i}$ mit der Metrik $m_{A,B}^{i,d,i}$ kleiner als die Summe der zugehörigen MRPM-Werte, so dass die Gleichung in (2) erfüllt ist, dann muss ein Simple-Loop vorliegen.

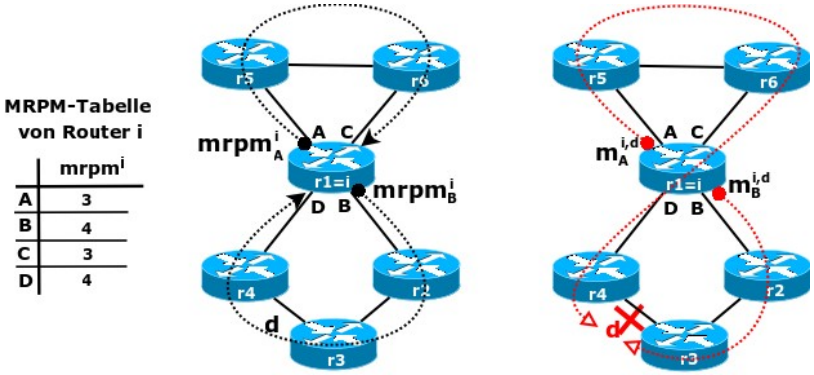


Abbildung 3.14: Der X-Test erkennt X-Kombinationen

Für das Beispiel in Abbildung 3.14 gelten die Metriken: $3+4>5+3-1(f)$.
Somit wird hier kein Simple-Loop erkannt und eine alternative Route $P_A^{i,d}$ mit der Metrik $m_A^{i,d}=5$ würde in diesem Fall abgelehnt werden.
Mit der Aktivierung des Split-Horizon und der Anwendung des X- und Y-Tests können alle möglichen Source-Loops erkannt und vermieden werden. Die Routen-Kombinationen, die nicht als Source-Loop erkannt werden, sind Simple-Loops und die Voraussetzung dafür, dass der Router, bei Ausfall einer vorhandenen Route, eine alternative Route zum Subnetz über ein anderes Interface empfangen kann.

3.3 Vermeidung des Counting-to-Infinity Problems

Die Theorie des RIP-MTI-Algorithmus besagt, dass das Counting-to-Infinity Problem einen Routing-Loop voraussetzt und vermieden werden kann, wenn Source-Loops im Netzwerk verhindert werden. Dabei stehen Source-Loops, Routing-Loops und das Counting-to-Infinity Problem in folgendem Zusammenhang (Beweis in [Sch99] S. 46 ff.):

Das Counting-to-Infinity Problem setzt einen Routing-Loop voraus. Ein Routing-Loop ist eine Route, die auf dem Weg zum Subnetz d einen Router $i \in R$ mindestens zweimal durchläuft. Aus Sicht des Routers i beschreibt die Route bis vor dem zweiten Durchlauf von Router i einen Source-Loop. Wenn Router i diesen Source-Loop verhindert, die Route also nicht in seine Routing-Tabelle einträgt, wird sie auch nicht an den nachfolgenden Router $i+1 \in R$ weitergeleitet. Somit wird die Route nicht weitergeführt und es kann kein Routing-Loop entstehen, womit auch das Counting-to-Infinity Problem nicht mehr auftreten kann.

1) Das Counting-to-Infinity Problem kann nicht auftreten, wenn kein Routing-Loop existiert.
2) Routing-Loops können nicht existieren, wenn Source-Loops verhindert werden.
3) Werden Source-Loops lokal, von jedem RIP-Router, verhindert, so kann auch das Counting-to-Infinity Problem global, im gesamten Netzwerk, nicht mehr auftreten.

Der von A. Schmid vorgestellte RIP-MTI-Algorithmus kann in zwei Arbeitsphasen, Simple-Loop-Erkennung und Source-Loop-Verhinderung, unterteilt werden. Welche Phase für jede einzelne Route in der Routing-Tabelle aktiv ist, hängt von der Metrik der Route, bzw. deren Gültigkeit, ab. Ist die Route in der Routing-Tabelle gültig und ihre Metrik somit besser als RIP-Infinity, so überprüft der RIP-MTI-Algorithmus alle über ein anderes Interface eintreffenden, alternativen Routen zum gleichen Subnetz mit der gleichen oder einer schlechteren Metrik, ob die Routen-Kombination, aus einer dieser alternativen Routen und der in der Routing-Tabelle eingetragenen Route zum Subnetz, den X- und Y-Test besteht und somit ein Simple-Loop zwischen den beiden Interfaces existiert. Der Simple-Loop mit der besten Metrik wird als MSILM-Metrik für die beiden Interfaces in die lokale MSILM-Tabelle eingetragen. Die kleinste MSILM-Metrik eines jeden Interface wird zusätzlich noch als MRPM-Metrik in einer eigenen MRPM-Tabelle gespeichert und wieder für die X- und Y-Tests verwendet. Ist die Metrik des erkannten Simple-Loops größer oder ist die Routen-Kombination kein Simple-Loop, so passiert in dieser Phase nichts weiter, da in der Routing-Tabelle eine gültige Route zum Subnetz vorhanden ist, die nur durch eine Route mit kürzerer Metrik ersetzt werden kann. Konventionelle RIP-Router würden alternative Routen mit gleicher oder schlechterer Metrik überhaupt nicht beachten, sondern solange die Route in ihrer Routing-Tabelle gültig ist sofort verwerfen.

Fällt die in der Routing-Tabelle eingetragene Route zum Subnetz nun aus und erhält damit die Metrik RIP-Infinity (16), so wechselt der RIP-MTI-Algorithmus für diese Route in die Phase der Source-Loop-Verhinderung. In dieser Phase greift der RIP-MTI-Algorithmus auf die zuvor gesammelten Information aus der MSILM-Tabelle zurück. Trifft hier nun eine alternative Route zum gleichen Subnetz über ein anderes Interface ein, so errechnet der RIP-MTI-Algorithmus die Metrik der Routen-Kombination aus dieser und der letzten gültigen Metrik der ausgefallenen Route. Ist die Metrik dieser Routen-Kombination, aus alter, ausgefallener und neuer, alternativer Route, größer als der vorhandene MSILM-Wert der zugehörigen Interfaces, dann wird die neue, alternative Route in die Routing-Tabelle aufgenommen. Ist die Metrik der Routen-Kombination kleiner als der MSILM-Wert, dann wird die neu angebotene, alternative Route verworfen, da sie keine gültige Route zum Subnetz darstellt, sondern höchstwahrscheinlich in einer anliegenden Netzwerkschleife entstand. Damit die neue, alternative Route zum Subnetz d über Interface A durch den RIP-MTI-Algorithmus nicht abgelehnt wird, muss also gelten: $m_A^{i,d} + m_B^{i,d} - 1 \geq msilm_{A,B}$

Die Metrik der Routen-Kombination, aus der letzten gültigen und der alternativen Route, kann nur dann kleiner sein als der vorhandene MSILM-Wert, der beiden zugehörigen Interfaces, wenn zuvor in der Simple-Loop-Erkennungs-Phase kein Simple-Loop zwischen den entsprechenden Interfaces erkannt wurde und der MSILM-Wert hier noch seinen initialen Wert RIP-MTI-Infinity (31) besitzt. Jede Routen-Kombination aus zwei Routen mit einer Metrik kleiner RIP-Infinity (16) hat eine bessere Metrik als RIP-MTI-Infinity. Wurde zuvor ein Simple-Loop zwischen den zugehörigen Interfaces erkannt, so besitzt jede Routen-Kombination, die auf eine gültige, alternative Route schließen lässt, eine gleiche oder schlechtere Metrik als der vorhandene MSILM-Wert der zugehörigen Interfaces. Die Arbeitsweise des RIP-MTI-Algorithmus lässt sich verkürzt über die folgende Regel beschreiben:

> *Wird nach dem Ausfall einer Route eine alternative Route über ein anderes Interface angeboten und existiert zwischen dem Interface dieser alternativen Route und dem der ausgefallenen Route ein Simple-Loop, so ist die alternative Route gültig und in die Routing-Tabelle aufzunehmen. Existiert jedoch kein Simple-Loop zwischen den beiden Interfaces der Routen, so handelt es sich bei der alternativen Route um das Ergebnis eines Source-Loops und sie ist abzulehnen.*

Diese Arbeitsweise des RIP-MTI-Algorithmus, wie sie ursprünglich von A. Schmid in [Sch99] vorgeschlagen und hier soweit vorgestellt wurde, wird im weiteren Verlauf der Ausarbeitung als „RIP-MTI-Normal-Mode“ bezeichnet. Der RIP-MTI-Normal-Mode arbeitet sehr gut in Netzwerken mit einer einfachen Topologie, in der die Router maximal zwei Interfaces besitzen, die in der selben Netzwerkschleife liegen. Liegen mehr als zwei Interfaces eines Routers in einer Netzwerkschleife, bzw. ist ein Interface eines RIP-MTI-Routers mehr als einmal in dessen lokaler MSILM-Tabelle aufgeführt, so handelt es sich hierbei um sogenannte „verschachtelte“ Netzwerkschleifen (nested loops), die nur in stark vermaschten Topologien vorkommen. Damit der RIP-MTI-Algorithmus auch in solchen komplexeren Topologien das Counting-to-Infintiy Problem verhindern kann, müssen einige Korrekturen am Konzept vorgenommen werden, die in den nachfolgenden Kapiteln beschrieben sind.

3.4 Korrekturen des RIP-MTI-Algorithmus

Tests mit verschiedenen Implementierungen des RIP-MTI-Algorithmus haben einige konzeptionelle Schwächen bei der Verhinderung des CTI-Problems durch den bis hier vorgestellten RIP-MTI-Algorithmus aufgezeigt, so dass unter bestimmten Umständen auch weiterhin das CTI-Problem auftreten kann. Im folgenden sind einige Korrekturen am Konzept des RIP-MTI-Algorithmus erläutert, die diese Schwächen beheben sollen.

3.4.1 Probleme in stark vermaschten Netzwerk-Topologien

Nach dem bisherigen Konzept des RIP-MTI-Algorithmus (RIP-MTI-Normal-Mode) wird versucht alle am Router anliegenden Simple-Loops zu erkennen und nach deren Vorkommen zu entscheiden, ob eine ausgefallene Route durch die eingetroffene, alternative Route ersetzt werden darf. Wurde zwischen dem Interface der ausgefallenen Route und dem Interface der alternativen Route ein Simple-Loop erkannt, dann darf die alternative Route die ausgefallene Route in der Routing-Tabelle ersetzen. Wurde jedoch kein Simple-Loop zwischen den beiden Interfaces erkannt, dann wird die alternativ angebotene Route als Ergebnis eines Source-Loops angesehen und abgelehnt. Diese Vorgehensweise ist allerdings nur zum Teil korrekt. Existiert kein Simple-Loop zwischen zwei Interfaces, so kann auch kein Routing-Update mit einer alternativen Route über das jeweilig andere Interface eintreffen und falls doch eine alternative Route eintrifft, dann ist diese Höchstwahrscheinlich[6] ein Source-Loop, dessen Aufnahme in die Routing-Tabelle das CTI-Problem verursachen kann. Der Umkehrschluss jedoch, dass dort wo ein Simple-Loop zwischen zwei Interfaces vorhanden ist auch kein Source-Loop auftreten kann, sondern nur gültige, alternative Routen eintreffen, gilt nicht, wie T. Kleeman in seiner Diplomarbeit [Kle01] S.37 ff. zeigt. Das CTI-Problem kann hier nach wie vor auftreten. Somit funktioniert der RIP-MTI-Normal-Mode nur in solchen Topologien, in denen der Source-Router i, der den Source-Loop

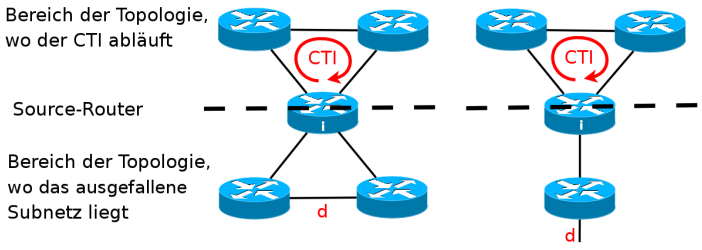


Abbildung 3.15: Topologien, in denen das bisherige Konzept des RIP-MTI-Normal-Mode funktioniert

6 Sofern zuvor keine (Transfer)-Subnetze vom Routing-Vorgang gefiltert wurden, was das erkennen von Simple-Loops behindern würde

erkennen und verwerfen soll, die einzige Verbindung zwischen dem Subnetz der ausgefallenen Route und der Netzwerkschleife, in der das CTI-Problem auftreten könnte, in der gesamten Topologie ist. Bei den Topologien in Abbildung 3.15 ist dies der Fall. Der Source-Router i trennt die beiden Bereiche der Topologie klar voneinander ab und jede Route zum Subnetz d führt für die Router der Netzwerkschleife über Router i. Gibt es allerdings eine weitere Verbindung zum Subnetz d, über einen anderen Router der Netzwerkschleife, dann genügt das bisherige Konzept nicht mehr denn das CTI-Problem kann weiterhin auftreten.

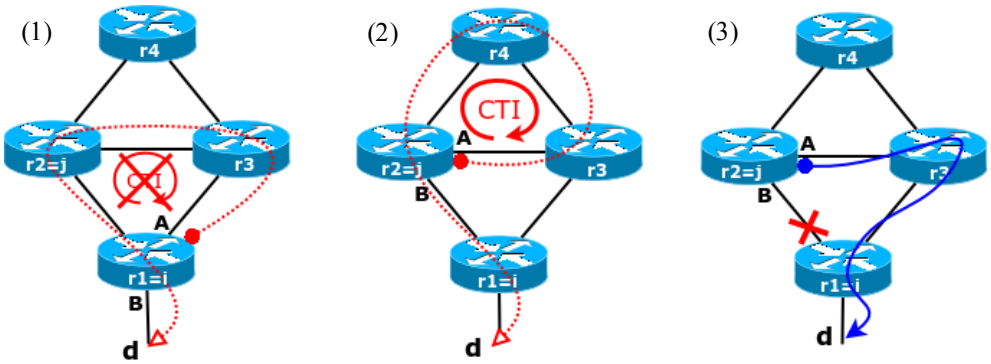


Abbildung 3.16: Erweiterte Y-Topologie mit verschachtelten Netzwerkschleifen (nested loops)

Die erweiterte Y-Topologie in Abbildung 3.16 ist ein Beispiel für eine Topologie mit verschachtelten Netzwerkschleifen, in der der RIP-MTI-Normal-Mode kein ausreichendes Kriterium mehr darstellt, um das Auftreten des CTI-Problems gänzlich zu verhindern.
Nach wie vor kann das CTI-Problem durch den RIP-MTI-Normal-Mode dort in der Topologie verhindert werden, wo das Subnetz allein durch den Source-Router von der Netzwerkschleife getrennt ist, bzw. der Source-Router der einzige Anbieter einer Route zum Subnetz für die anderen Router der Netzwerkschleife ist.
In Abbildung 3.16 (1) ist Router i die einzige Verbindung zwischen der Netzwerkschleife über den Routern i, r2 und r3 und dem Subnetz d. Das CTI-Problem wird hier durch einen RIP-MTI-Normal-Mode in Router i vermieden, der den Source-Loop erkennen und verhindern kann. Router i erkennt hier keinen Simple-Loop zwischen den beiden Interfaces A und B und lehnt deshalb eine über Interface A angebotene, alternative Route zum Subnetz d ab.
In der Topologie kann allerdings nach wie vor das CTI-Problem über der Route zum Subnetz d auftreten. In Abbildung 3.16 (2) kann Router j einen Source-Loop, der sich über die Router r3 und r4 bildet, nicht allein mit Hilfe des RIP-MTI-Normal-Mode erkennen und somit auch das CTI-Problem nicht verhindern. Router j erkennt einen Simple-Loop zwischen den Interfaces A und B und vermutet hinter der eintreffenden Route zum Subnetz d über Interface A eine gültige alternative Route. Diese Vermutung könnte auch durchaus zutreffen, wie in Abbildung 3.16 (3) dargestellt ist.
Auf dieses Problem des RIP-MTI-Normal-Mode weißt T. Kleemann in seiner Diplomarbeit [Kle01] hin. Als Lösung sieht er eine wesentlich strengere Bewertung der eingehenden alternativen Routen vor, um die Source-Loops zu erkennen und die alternative Route nicht als Ersatz für die Ausgefallene aufzunehmen. Die Arbeitsweise des RIP-MTI-Algorithmus bei der Simple-Loop Erkennung, während eine gültigen Route in der Routing-Tabelle ist, bleibt dabei unverändert. Die Veränderung der Arbeitsweise des RIP-MTI-Algorithmus nach T. Kleemann wird im weiteren Verlauf der Arbeit „RIP-MTI-Strict-Mode“ genannt.
Bislang hat der RIP-MTI-Normal-Mode alle alternativen Routen als Ersatz für eine ausgefallene Route akzeptiert, deren Routen-Kombination mit der ausgefallenen Route mindestens die Länge des Simple-Loops zwischen den zugehörigen Interfaces betrug. Deshalb

kann der Source-Loop in Abbildung 3.16 (2) nicht erkannt werden und wird für eine gültige Route gehalten. Für Router j ist die Situation in Abbildung 3.16 (2) daher wie die in 3.16 (3).

Zu Abbildung 3.16 (1) :
Router i erkennt den Source-Loop: $m_A^{i,d} + m_B^{i,d} - 1 > msilm_{A,B}^{i} \;\rightarrow\; 4+1-1>31\,(f)$

Zu Abbildung 3.16 (2) :
Router j erkennt den Source-Loop nicht: $m_A^{j,d} + m_B^{j,d} - 1 > msilm_{A,B}^{j} \;\rightarrow\; 5+2-1>3$

Zu Abbildung 3.16 (3) :
Router j erkennt die gültige, alternative Route: $m_A^{j,d} + m_B^{j,d} - 1 > msilm_{A,B}^{j} \;\rightarrow\; 3+2-1>3$

Nach der Arbeitsweise des RIP-MTI-Strict-Mode wird der X- und Y-Test auch während der Phase der Source-Loop Verhinderung angewandt. Somit genügt nicht mehr nur ein vorhandener Simple-Loop zwischen den beiden Interfaces, sondern zusätzlich muss die Kombination der neuen alternativen Route und der alten ausgefallenen Route ein Simple-Loop ergeben. Auf dem Pfad dieses Simple-Loops liegt das Subnetz, sofern es nach dem Ausfall der vorhandenen Route überhaupt noch erreichbar und die alternative Route gültig ist.
Verwendet Router j in Abbildung 3.16 (2) den X- und den Y-Test auch in der Phase der Source-Loop Verhinderung, dann kann der hier vorliegende Source-Loop erkannt werden. Genau genommen wird hier dann kein Simple-Loop aus alternativer und ausgefallener Route über dem Subnetz d erkannt.

Y-Test : $mrpm_A^{j} + m_B^{j,d} > m_A^{j,d} \;\rightarrow\; 3+2>5\,(f)$

X-Test : $mrpm_A^{j} + mrpm_B^{j} > m_A^{j,d} + m_B^{j,d} - 1 \;\rightarrow\; 3+3>5+2-1\,(f)$

Der RIP-MTI-Strict-Mode ist allerdings viel zu Streng in seiner Bewertung der alternativen Routen, was dazu führt, dass auch viele gültige, alternative Routen abgelehnt werden können, weil die Kombination der beiden Routen für den X-/Y-Test keinen Simple-Loop ergibt. Das hängt damit zusammen, dass X- und Y-Test jeweils auf den MRPM-Werten, den Metriken der minimalen Simple-Loops, der beiden Interfaces aufbauen. Für den Y-Test bedeutet das letztlich, dass eine angebotene, alternative Route immer entlang des kleinsten Simple-Loops des zugehörigen Interfaces verlaufen müsste, um den Y-Test zu bestehen. Das ist in (3) der Fall, muss allerdings nicht immer so sein, wie in Kapitel 5.5.2 an einem Beispiel gezeigt wird.
Beim X-Test ist das Problem, dass er sich an den MRPM-Werten der minimalen Simple-Loops der zugehörigen Interfaces orientiert. Hier werden alle alternativen Routen durch den X-Test abgelehnt, die zu Subnetzen führen die weiter entfernt vom Router weg liegen und deren Routen daher generell eine höhere Metrik besitzen, als die kleinen MRPM-Werte der zugehörigen Interfaces überhaupt erlauben. Nur die gültigen, alternativen Routen zu Subnetzen in der näheren Umgebung akzeptiert der X-Test.
Das Problem ist, dass durch dieses Verhalten des RIP-MTI-Strict-Mode die Konvergenz des Netzwerks, nach Ausfall einer Route, im Vergleich zu konventionellen RIP-Routern, erheblich verzögert wird. Der RIP-MTI-Strict-Mode lehnt die gültige, alternative Route solange ab, wie er Zugriff auf die ausgefallene Route in der Routing-Tabelle hat. Erst nach Ablauf des Garbage-Collector-Timers (Standard: nach 120 Sekunden) und der damit verbundenen vollständigen Löschung der alten Route aus der Routing-Tabelle, kann die neue alternative Route akzeptiert werden, da sie nicht mehr durch den RIP-MTI-Strict-Mode überprüft wird. Sie ist dann keine alternative Route mehr und ersetzt auch nicht die ausgefallene Route, sondern wird als vollständig neuer Eintrag in die Routing-Tabelle angesehen.

Um diesen Nachteil des RIP-MTI-Strict-Mode entgegenzuwirken, wurden verschiedene Erweiterungen für den RIP-MTI-Algorithmus ausgearbeitet, die unter dem Begriff „RIP-MTI-Careful-Mode“ zusammengefasst sind. Auch die RIP-MTI-Careful-Mode Erweiterungen verändern die Arbeitsweise des ursprünglichen RIP-MTI-Algorithmus nur in der Phase der Source-Loop Verhinderung. Die Phase der Simple-Loop Erkennung wird nicht verändert.
Der RIP-MTI-Careful-Mode basiert direkt auf dem RIP-MTI-Strict-Mode, allerdings markiert der Careful-Mode, zum Zwecke der Wiedererkennung, die ausgefallene Route in der Routing-Tabelle, wenn ihre angebotene, alternative Route den X-/Y-Test nicht besteht und abgelehnt wird. Zusätzlich sendet der RIP-MTI-Careful-Mode nach der Ablehnung einer alternativen Route immer ein Routing-Update aus, das die ausgefallene Route zum Subnetz mit Metrik RIP-Infinity enthält, um einen eventuell über der Route entstandenen Source-Loop auch gleich bei den benachbarten Routern der Netzwerkschleife zu löschen. Wird die alternative Route erneut angeboten, so kann dies anhand der Markierung an der ausgefallenen Route aus der Routing-Tabelle erkannt werden. Es ist sehr unwahrscheinlich, dass ein Source-Loop lange genug und über mehrere Update-Intervalle existiert, um mehrmals einem Router angeboten zu werden. Der Source-Loop wird für gewöhnlich mit dem nächsten ausgesendeten Routing-Update des Source-Routers gelöscht. Um zu Überprüfen ob eine alternative Route eine beständige, gültige Route oder das Ergebnis eines unbeständigen Source-Loops ist, wurden in dieser Arbeit für den RIP-MTI-Careful-Mode drei Erweiterungen geschaffen, die hier kurz aufgelistet werden.

1) RIP-MTI-Careful-DT-Mode (Deny-Timer): Diese RIP-MTI-Careful-Mode Variante verwirft eine abgelehnte Route nicht einfach, sondern speichert sie lokal zwischen. Wenn die abgelehnte Route über eine bestimmte Zeitspanne hinweg nicht über ein eingehendes Routing-Update mit Metrik RIP-Infinity widerrufen wird, dann wird sie in die Routing-Tabelle übernommen.
2) RIP-MTI-Careful-RT-Mode (Request-Timer): Dieser RIP-MTI-Mode funktioniert ähnlich wie der RIP-MTI-Careful-DT-Mode, nur das die verworfene Route nicht lokal im RIP-MTI-Router zwischengespeichert, sondern verworfen wird. Nach Ablauf einer kurzen Zeitspanne wird dann ein Request an die benachbarten Router gesendet, um die Route erneut zu erhalten. Die empfangene Route wird dann, ohne weitere Überprüfung durch den RIP-MTI-Algorithmus, in die Routing-Tabelle übernommen.
3) RIP-MTI-Careful-ESHC-Mode (External-Split-Horizon-Check): Als eine weitere Lösung wurde diese RIP-MTI-Careful-Mode Variante entwickelt. Bei Ablehnung einer alternativen Route, speichert diese Variante eine Liste mit allen Interfaces, die einen Simple-Loop, mit dem zur abgelehnten Route gehörenden Interface, besitzen. Erst wenn auch über diese verbundenen Interfaces eine alternative Route zum Subnetz eingetroffen ist, was aufgrund der Simple-Loop Verbindungen der Fall sein müsste, wird die alternative Route in die Routing-Tabelle aufgenommen um die ausgefallene Route zu ersetzen.

Vorweg kann hier schon der RIP-MTI-Careful-RT-Mode als möglicherweise bester unter den entworfenen Careful-Modes hervorgehoben werden. Im Vergleich zu den anderen arbeitet er sehr viel effizienter und ist weit weniger kompliziert in seiner Implementierung. Liegen keine verschachtelten Schleifen am RIP-MTI-Router an, dann kann jedoch weiterhin der RIP-MTI-Normal-Mode verwendet werden.

3.4.2 Reduktion des RIP-MTI-Algorithmus auf den Y-Test

Der RIP-MTI-Strict-Mode verwendet den X- und Y-Test nicht nur um Simple-Loops während der konvergenten Phase des Netzwerks zu erkennen, sondern auch nach Ausfall einer Route um Source-Loops zu verhindern und das CTI-Problem so zu vermeiden. Problematisch ist dabei die Verwendung des X-Tests, der alternative Routen zu Subnetzen, die weiter entfernt vom Router liegen und darum generell eine höhere Metrik besitzen, durchweg ablehnt, wenn die MRPM-Werte der beiden beteiligten Interfaces zu klein sind.

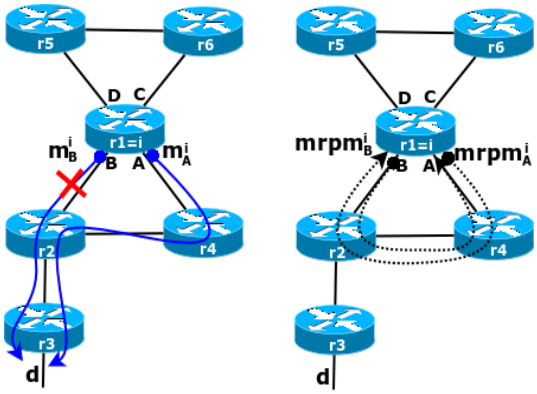


Abbildung 3.17: Der X-Test lehnt die gültige, alternative Route ab

Im Beispiel in Abbildung 3.17 berechnen X-Test und Y-Test folgendes:

X-Test : $mrpm^i_A + mrpm^i_B > m^{i,d}_A + m^{i,d}_B - 1 \;\rightarrow\; 3+3>4+3-1\,(f)$

Y-Test : $mrpm^i_A + m^{i,d}_B > m^{i,d}_A \;\rightarrow\; 3+3>4$

Der X-Test ist hier nicht bestanden und die eigentlich gültige, alternative Route über Interface A wird darum abgelehnt. Dementsprechend würde der RIP-MTI-Strict-Mode gerade in größeren, stark vermaschten Netzwerk-Topologien sehr viele gültige, alternative Routen ablehnen und so die Konvergenz des Netzwerks spürbar verlangsamen. Wird bedacht, dass die Metrik $m^{i,d}_A$ einer alternativen Route $P^{i,d}_A$ immer größer oder gleich der Metrik $m^{i,d}_B$ der ausgefallenen Route $P^{i,d}_B$ ist, da RIP immer die Route mit der besseren Metrik in die Routing-Tabelle einträgt, also gilt $m^{i,d}_A \geq m^{i,d}_B$, dann würde der X-Test generell alle alternativen Routen als Ersatz für Ausgefallene ablehnen, wenn die Summe der MRPM-Werte der Interfaces kleiner oder gleich der Routen-Kombination der Route mit sich selbst ist, also gilt $mrpm^i_A + mrpm^i_B \leq 2 * m^{i,d}_B - 1$. Zwar lehnt auch der Y-Test so manche gültige, alternative Route ab, allerdings nicht im gleichen Ausmaß wie der X-Test. Die bisher entworfenen RIP-MTI-Careful-Mode Varianten können dieses Problem lediglich lindern, indem sie die Dauer der Ablehnung einer gültigen, alternativen Route reduzieren. Das Problem des Ablehnens gänzlich lösen können sie allerdings nicht. Der X-Test kann jedoch vollständig durch den Y-Test ersetzt werden, was die Zahl der falsch, abgelehnten Routen stark verringert. Bislang kann der Y-Test nur die Y-Kombinationen unter den Source-Loops erkennen. Dabei muss die Metrik der alternativen Route $m^{i,d}_A$ auf der letzten gültigen Metrik der ausgefallenen Route $m^{i,d}_B$ und dem am Interface anliegenden minimalen Simple-Loop $mrpm_A$ basieren. Ist das nicht der Fall, weil der Router nach Ausfall der bisherigen Route zum Subnetz für kurze Zeit eine alternative Route aus anderer Richtung erhalten hat, die gleich darauf aber ebenfalls ausgefallen ist, so liegt eine X-Kombination vor und die Erkennung des Source-Loops gelingt nur mit dem X-Test, da die zuvor ausgefallene Route, auf der der Source-Loop aufbaut, für den Router nicht mehr verfügbar ist. Sie gelingt aber auch dem Y-Test, wenn die Metrik dieser letzten, kurzzeitig angenommen Route gleich oder sogar kleiner der Metrik der Route ist, auf der der Source-Loop basiert. Speichert der RIP-MTI-Algorithmus

bei einem Ausfall der Route nicht wie bisher, einfach nur die zuletzt gültige Route für den Vergleich mit einer alternativen Route über den Y-Test, sondern unter den, in einer gewissen Zeitspanne zuletzt gültigen Routen, die Route mit der kleinsten Metrik, dann kann er die X-Kombinationen als einfache Y-Kombinationen erkennen. Diese Verzögerung der alten Metrik, bzw. dieses längere Speichern von Routen mit kleinerer Metrik für die ausgefallene Route $m_B^{i,d}$ im RIP-MTI-Algorithmus wird im weiteren Verlauf der Arbeit als „oldmetric_delay“ bezeichnet, bzw. als „verzögerter Y-Test“.
Da der RIP-MTI-Algorithmus auch nicht aktiv wird, wenn über ein Routing-Update, welches die bisherige Route bestätigt, über das gleiche Interface eintrifft, wird neben der Metrik auch das zugehörige Interface gespeichert und zum Vergleich herangezogen, um Probleme von dieser Seite zu vermeiden.
In Abbildung 3.18 ist die Problematik dieses Falls dargestellt. In (1) besitzt jeder Router des Netzwerks eine Route zum Subnetz d. In (2) ist das Subnetz d nun nicht mehr zu erreichen. Nachdem allerdings Router r2 die Unerreichbarkeit des Subnetzes registriert hat, erhält er noch ein periodisches Routing-Update von Router r3, der allerdings noch die ältere, mittlerweile ungültige Information der Erreichbarkeit besitzt. Router r2 nimmt die Route zum Subnetz d über r3 an, da es sich für ihn um eine gültige, alternative Route aufgrund des vorhandenen Simple-Loops handelt. Nachdem r3 die Unerreichbarkeit des Subnetzes d erkannt hat, ist die Topologie anfällig für ein, von r4 ausgehendes, CTI-Problem (3). Router r2 kann nun mit der bisherigen Implementierung des RIP-MTI-Algorithmus den Source-Loop nicht erkennen und verhindern, da die Route über das gleiche Interface, wie die Vorherige angeboten wird, bzw. vom gleichen Nachbarrouter r3 kommt. Als ein weiteres Problem kommt hinzu, dass die bisherige Trennung in zwei Phasen, von Simple-Loop Erkennung und Source-Loop Verhinderung, hier nun nicht mehr gelten kann, da der Zustand der Route zum Subnetz d von r2 nun keine Rolle mehr spielt. Router r2 muss den Source-Loop auch erkennen, wenn er eine gültige Route zum Subnetz in der Routing-Tabelle hat.

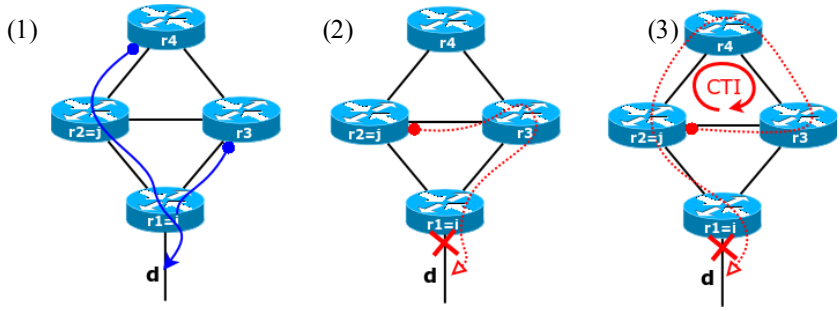


Abbildung 3.18: Problem mit falschen Routing-Updates über das gleiche Interface

Gelöst werden kann dieses Problem, über eine verzögerte Betrachtung der vorhandenen Routing-Informationen durch den RIP-MTI-Algorithmus, was bedeutet, dass auch eine kurze Zeit nachdem eine ausgefallene Route zu einem Subnetz wieder über eine gültige Metrik verfügt, eine weitere Veränderung der Metrik mit den noch älteren Routing-Informationen überprüft werden muss. In diesem Fall muss also in Abbildung 3.18 (3) Router r2 die angebotene Route über r3 trotz vorhandener, gültiger Route mit dem verzögerten Y-Test und den Routing-Informationen der Route zum Subnetz d, die in der letzten Zeit die kleinste Metrik hatte, überprüfen. Ist das Ergebnis wie hier negativ, so muss die gültige Route in der Routing-Tabelle für ungültig erklärt und auf Metrik RIP-Infinity (16) gesetzt werden. Der Y-Test berechnet hier das folgende: $mrpm_A^j + m_B^{j,d} > m_A^{j,d} \rightarrow 3+2>5\,(f)$
Diese Unsicherheiten existieren allerdings immer nur für eine kurze Zeit nach einem Ausfall, weswegen die verzögerte Betrachtung ebenfalls nur kurzzeitig vorgehalten werden muss.

3.4.3 Problem der Vollständigkeit der MSILM-Tabelle

Die MSILM-Tabelle beinhaltet jeweils die Metrik des minimalen Simple-Loops zwischen zwei Interfaces. Der RIP-MTI-Algorithmus erkennt einen Simple-Loop zwischen zwei Interfaces über das jeweilige Eintreffen eines Routing-Update mit einer Route zum gleichen Subnetz. Es gibt jedoch Netzwerk-Topologien in denen der RIP-MTI-Algorithmus nicht alle vorhandenen Simple-Loops erkennen kann, da in der Topologie kein Subnetz existiert, zu dem über jedes Interface des Routers, über dem dies eigentlich möglich wäre, eine Route zum gleichen Subnetz angeboten wird.
Besitzt ein Router mehrere Interfaces, die alle über eine Netzwerkschleife miteinander in Verbindung stehen, so kann es bei entsprechend stark vermaschter Topologie dazu kommen, dass der Router einige kürzere Abschnitte der Netzwerkschleife und somit einige Simple-Loops zwischen zwei Interfaces nicht erkennen kann. Betroffen sind hierbei vor allem die größeren Simple-Loops zwischen zwei „entfernter liegenden“ Interfaces, die kleinere Simple-Loops, zwischen zwei „näher liegenden“ Interfaces, umspannen.

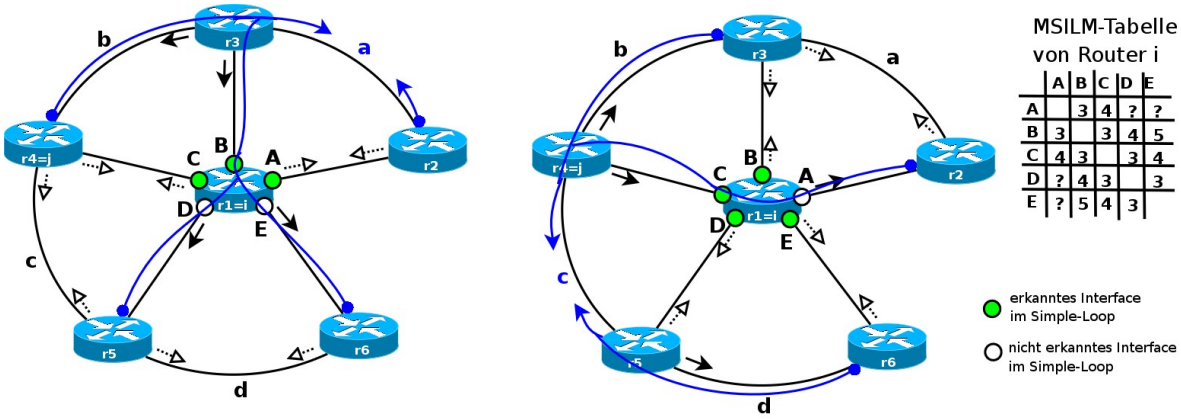


Abbildung 3.19: Nicht immer können alle Simple-Loops zwischen den Interfaces gefunden werden

Abbildung 3.19 zeigt eine solche Topologie. Der Simple-Loop zwischen den Interfaces A und E, sowie auch zwischen A und D, des Routers i kann nicht vom RIP-MTI-Algorithmus erkannt werden, da in der ganze Topologie kein Subnetz existiert, zu dem eine Route über dem Interface A und eine Route über dem Interface E des Router i existiert. Da allerdings jeweils eine Route zum Subnetz a über die Interfaces A und C des Routers i existiert und eine Route zum Subnetz d über die Interfaces C und E, könnte der RIP-MTI-Algorithmus anhand der Einträge des Interfaces C in seiner lokalen MSILM-Tabelle erkennen, das ein Simple-Loop auch zwischen den Interfaces A und E vorhanden ist, da beide mit Interface C in Verbindung stehen.
Abbildung 3.20 zeigt die beiden kleineren Simple-Loops zwischen A und C sowie zwischen C und E und den daraus kombinierten größeren Simple-Loop zwischen A und E, der allein mit Hilfe eingehender Routing-Updates nicht erkannt werden würde. Weiterhin könnte auch der MSILM-Wert zwar nicht genau berechnet, aber doch zumindest abgeschätzt werden. Dafür werden die erkannten MSILM-Werte der Interfaces A und

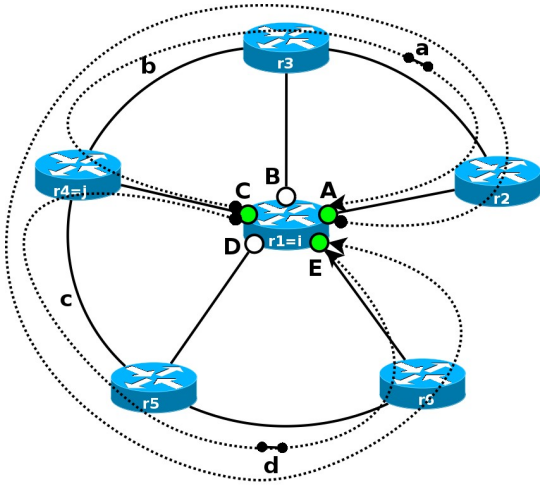


Abbildung 3.20: Berechnung eines Simple-Loop

C sowie C und E addiert. Von dieser errechneten Metrik muss allerdings noch die Metrik $m_C^{i,j}$ des Pfades $P_C^{i,j}$ zwischen dem Router i und j abgezogen werden. Bei Router j treffen sich die beiden Simple-Loops und beschreiben den selben Pfad zum Router i. Die Metrik des Pfades zwischen Router i und j beträgt mindestens ein Hop, ist jedoch für Router i letztlich unbekannt, weswegen auch die Metrik des größeren Simple-Loops zwischen den Interfaces A und E nur abgeschätzt werden kann.

$$msilm_{A,E}^{i} \leq msilm_{C,A}^{i} - m_C^{i,j} + msilm_{C,E}^{i} - m_C^{i,j}$$
$$\Leftrightarrow msilm_{A,E}^{i} \leq msilm_{C,A}^{i} + msilm_{C,E}^{i} - 2 * m_C^{i,j}$$

In der obigen Beispiel-Topologie würde somit folgendes berechnet werden:

$$msilm_{A,E}^{i} \leq 4 + 4 - 2 * 1$$
$$\Rightarrow msilm_{A,E}^{i} \leq 6$$

Mit Hilfe dieser Gleichung kann die MSILM-Tabelle vervollständigt werden. Die berechneten MSILM-Werte sind zwar nur abgeschätzt und eventuell größer als die Metrik des tatsächlich vorhandenen Simple-Loops, doch trotzdem wesentlich aussagekräftiger gegenüber dem initialen Wert RIP-MTI-Infinity und einem nicht erkannten Simple-Loop.
Wirklich betroffen von diesem Problem ist allerdings lediglich der RIP-MTI-Normal-Mode, sowie der RIP-MTI-Careful-ESHC-Mode. Über Tests mit letzterem wurde dieses Problem festgestellt, da der Careful-ESHC eine alternative Route, über alle, mit einem Simple-Loop in Verbindung stehenden, Interfaces bestätigt haben möchte bevor er die Route akzeptiert. In diesem Fall können aber nicht alle miteinander verbundenen Interfaces erkannt werden, was dem CTI-Problem eine Lücke lässt. Der RIP-MTI-Strict-Mode und die beiden RIP-MTI-Careful-Modes DT und RT sind nicht von dem Problem betroffen, da sie theoretisch die MSILM-Werte nicht mehr benötigen und allein mit den MRPM-Werten funktionieren würden.

3.4.4 Orientierung über die Learned-From-IP-Adresse

Bisher galt, dass der RIP-MTI-Algorithmus das Counting-to-Infinity Problem verhindert, in dem er die vorhandenen Simple-Loops zwischen den Interfaces des Routers erkennt. In diesem Fall unterscheiden sich die Routen anhand der Interfaces voneinander. Ob und wie viele unterschiedliche alternative Routen zu einem Subnetz einem Router letztlich angeboten werden ist allerdings nicht von der Anzahl der Interfaces abhängig, sondern von der Anzahl und Lage der direkt benachbarten Router. Es gibt Netzwerk-Topologien in denen Router über ein einziges Interface mit mehreren benachbarten Routern direkt verbunden sind, zum Beispiel wenn hinter dem Interface ein „Switch“ oder ein „Hub“ geschaltet ist, an dem die Router angeschlossen sind. Der Router könnte über jeden seiner benachbarten Router eine entsprechende Route zum Subnetz erhalten und somit mehrere, unterschiedliche Routen zum gleichen Subnetz über ein einziges Interface angeboten bekommen. Damit der RIP-MTI-Algorithmus auch hier alle Simple-Loops erkennen kann, um bei einem Ausfall einer Route einen Source-Loop zu vermeiden, genügt die bisherige Orientierung über das ausgehende Interface O ∈ IF eines Hops H= (O, s, I) nicht mehr. Das entscheidende Element für die Existenz einer Route ist der benachbarte Router, der die Route weiter verbreitet. Eine Orientierung über das eingehende Interface I ∈ IF des benachbarten Routers ermöglicht, aufgrund der Identifizierung der Router, auch die Unterscheidung der verschiedenen Routen. Als eingehendes Interface bietet sich hier auf der Vermittlungsschicht die IP-Adresse des benachbarten Routers an. Diese IP-Adresse ist in den, von den Routern versendeten, Routing-Updates als Learned-From-IP-Address angegeben.

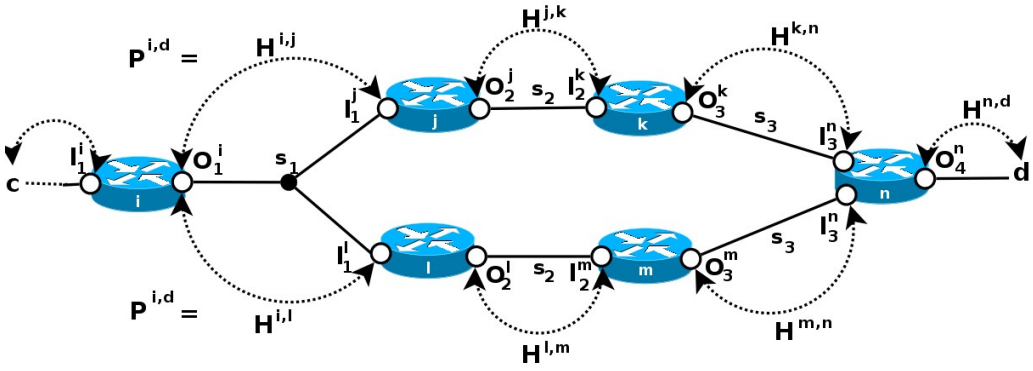


Abbildung 3.21: Zwei unterschiedliche Routen zum Subnetz d hinter dem gleichen Interface

Abbildung 3.21 zeigt eine Topologie in der hinter dem ausgehenden Interface O ∈ IF des Routers i die beiden Router j und l direkt benachbart sind, die dem Router i jeweils eine Route zum Subnetz d anbieten können. Hat Router i die Route über Router j übernommen und fällt diese Route aus, so bietet Router l direkt eine alternative Route an. Den Simple-Loop kann der RIP-MTI-Algorithmus des Routers i allerdings nicht erkennen, wenn er sich über sein ausgehendes Interface O orientiert.

Haben die Router j und l eine Route zum Subnetz c von Router i gelernt und fällt diese Route aus, so könnte keiner der Router einen Source-Loop vermeiden und das daraus folgende CTI-Problem verhindern. Mit einer Orientierung über die IP-Adresse der benachbarten Router gelingt dies dann allerdings.

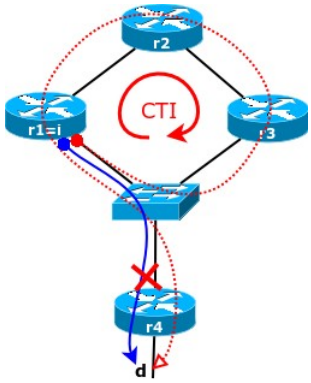


Abbildung 3.22: Topologie mit Switch

In Abbildung 3.22 ist eine Netzwerk-Topologie mit einem Switch dargestellt. Router r1 kann den Source-Loop nur erkennen und das CTI-Problem verhindern, wenn er die Router r3 und r4 über deren IP-Adresse voneinander unterscheiden kann. Nur so kann Router r1 auch die unterschiedlichen Routen, die jeweils über diese beiden Router verlaufen, voneinander unterscheiden und die Berechnungen zur Erkennung des Source-Loops durchführen.

Im Kapitel 5.7 wird der Nachteil der Orientierung über Interfaces noch einmal verdeutlicht und die Implementierung über die Learned-From-IP-Adresse vorgestellt. Die Orientierung über Interfaces genügt aber in Netzwerken, in denen hinter jedem Interface nur ein benachbarter Router zu finden ist.

Wie im bisherigen Verlauf dieser Ausarbeitung, so wird auch im weiteren Verlauf an der Orientierung über Interfaces bei den Beschreibungen und Erläuterungen der Einfachheit wegen festgehalten, auch wenn eine fertige Implementierung des RIP-MTI-Algorithmus sich unbedingt über die LF-IP-Adresse der benachbarten RIP-Router orientieren sollte.

3.5 Zusammenfassung

Der RIP-MTI-Algorithmus verhindert das CTI-Problem für das Routing Information Protocol (RIP) allein aufgrund einer erweiterten Auswertung der in den RIP-Routern vorhandenen Routing-Informationen. Somit wird die Kompatibilität zur Standard RIP-Spezifikation gewahrt. Der RIP-MTI-Algorithmus unterscheidet zwei Arten von Schleifen, Simple-Loops und Source-Loops, die aus zwei verschiedenen Routen zum gleichen Subnetz über unterschiedliche

Interfaces, zusammengesetzt sind. Dabei bestehen Source-Loops den X- und Y-Test nicht und können so von Simple-Loops unterschieden werden. Ein Simple-Loop entspricht im wesentlichen einer topologischen Netzwerkschleife auf der Vermittlungsschicht. Source-Loops wiederum sind Voraussetzung für das Auftreten des CTI-Problems.
Auf Basis des RIP-MTI-Algorithmus wurden in diesem Kapitel drei RIP-MTI-Modes vorgestellt. Der RIP-MTI-Normal-Mode erkennt Source-Loops nur in einfachen Netzwerk-Topologien. Damit ist er zwar schon eine Verbesserung gegenüber dem konventionellen RIP-Algorithmus, verhindert das CTI-Problem allerdings nicht immer. Der RIP-MTI-Strict-Mode arbeitet nach strengeren Kriterien und erkennt Source-Loops auch in komplexeren, vermaschten Topologien. Allerdings lehnt er unter Umständen auch gültige, alternative Routen als Ersatz für eine ausgefallene Route ab, was die Konvergenz des Netzwerks erheblich verzögern kann. Der RIP-MTI-Careful-Mode erweitert den RIP-MTI-Strict-Mode, mit dem Ziel die Konvergenz wieder zu beschleunigen. Der RIP-MTI-Careful-Mode versucht über verschiedene zusätzliche Kriterien in möglichst kurzer Zeit zu erkennen, ob eine alternative Route, die vom RIP-MTI-Strict-Mode abgelehnt wurde, auch tatsächlich das Ergebnis eines Source-Loops ist. Für den RIP-MTI-Careful-Mode wurden bislang drei verschiedene Varianten entwickelt, genannt ESHC, DT und RT. In dieser Ausarbeitung wird der RT (Request Timer) favorisiert, der eine zweifelhafte Route erstmal ablehnt, ein Routing-Update aussendet um den vermuteten Source-Loop in der Netzwerkschleife zu löschen und nach einer kurzen Zeitspanne eine Request-Nachricht sendet, um den Verbleib der Route beim Nachbarrouter zu überprüfen.
Vom RIP-MTI-Algorithmus, der in diesem Kapitel in der Theorie vorgestellt wurde, besteht bislang lediglich eine prototypische Implementierung, die noch nie in produktiven Systemen zum Einsatz gekommen ist. Einige der hier vorgestellten Korrekturen wurden erst durch die praktischen Erfahrungen mit diesem Prototyp aufgedeckt und es kann zum jetzigen Zeitpunkt noch nicht ausgeschlossen werden, dass dem RIP-MTI-Algorithmus noch weitere Veränderungen bevorstehen oder auch eine Veränderung wieder revidiert werden muss.
Auch wenn der RIP-MTI-Algorithmus zum Teil nicht mehr nach dem ursprünglich vorgestellten Konzept von A. Schmid aus [Sch99] funktioniert, konnte er bislang doch stets so erweitert werden, dass er nach wie vor das CTI-Problem verhindern kann, so dass die grundsätzliche Aussage, anhand erkannter topologischer Netzwerkschleifen das Entstehen von Routingschleifen und somit auch das CTI-Problem zu verhindern, sowie die aufgestellten, grundlegenden Berechnungen, nach wie vor gültig sind.

4 Die Software

Im folgenden Kapitel wird die in dieser Ausarbeitung verwendete und erweiterte Software vorgestellt. Als Basis für die Implementierung des RIP-MTI-Algorithmus dient der RIP-Routing-Daemon der Routing-Software-Suite Quagga. Die Funktionalität der RIP-MTI-Implementierung wurde in virtuellen Linux-Rechnernetzen getestet, die mit Hilfe der Virtualisierungssoftware VNUML aufgebaut wurden. Auch das an der Universität Koblenz entwickelte Steuerungs- und Analysewerkzeug XTPeer, mit dem das Verhalten der Quagga RIP-Daemons kontrolliert und protokolliert werden kann, wurde entsprechend an den RIP-MTI-Algorithmus angepasst.

4.1 Die Routing Software Suite Quagga

Quagga ist eine *IPv4/IPv6 Routing Software Suite*, die TCP/IP basiertes Routing unterstützt und verschiedene Routing-Dienstprogramme für die Verwendung unterschiedlicher IP-Routing-Protokolle, wie RIP, OSPF und BGP, unter UNIX-basierten Betriebssystemen, wie Linux, bereitstellt. Der Quellcode der Quagga Programme ist Open-Source und steht unter der GNU General Public Licence (GNU-GPL). Das Quagga-Projekt[7] ist eine Weiterführung des, ursprünglich von Kunihiro Ishiguro im Jahre 1996 gestarteten, GNU-Zebra Projektes, zielt aber auf eine breitere Unterstützung durch Freiwillige und Anwender. Quagga ist ein Programmpaket, bestehend aus mehreren eigenständigen Routing-Dienstprogrammen (Routing-Daemons), die unabhängig voneinander jeweils ein bestimmtes Routing-Protokoll unterstützten. So wird OSPFv2-Routing durch das Routing-Dienstprogramm „ospfd“ ermöglicht, „bgpd“ ermöglicht BGPv4-Routing und „ripd“ ist der RIP-Daemon, der Routing mittels RIP ermöglicht. Für alle Quagga Routing-Dienstprogramme ist der Core-Daemon „zebra“ die Schnittstelle zur, vom Linux-Kernel verwalteten Routing-Tabelle, über die Routen ausgetauscht werden können.

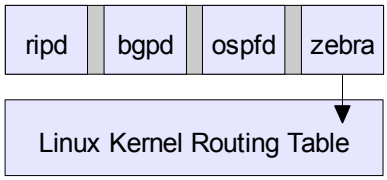


Abbildung 4.1: Quagga Architektur

Die Quagga Routing-Dienste können über eine jeweils eigene Konfigurationsdatei, die beim Start des Dienstes angegeben werden muss, sowie zur Laufzeit über eine Telnet-Schnittstelle, konfiguriert werden. Die Schnittstelle und die Kommandos für die Konfiguration sind an das Cisco IOS angelehnt. Neben der Konfiguration via Telnet existiert noch die integrierte Benutzerschnittstelle „vtysh“. Das „vtysh“-Terminal fungiert als eine Art Login-Proxy, über den sich ein Benutzer mit jedem lokal ausgeführten Quagga Routing-Dienst verbinden kann. Die einzelnen Quagga Routing-Dienste unterstützen auch das Netzwerk Management-Protokoll SNMP[8]. Über einen vorgeschalteten SNMP-Master-Agenten können entsprechende Status-Informationen via SNMP abgefragt werden.

Die in dieser Arbeit erstellte Implementierung des RIP-MTI-Algorithmus ist eine Erweiterung des RIP-Daemons aus der Quagga Routing-Suite in Version 0.99.4. Sie baut auf der Implementierung von T. Koch, aus [Koc05], auf. Die Implementierung von T. Koch leidet allerdings unter den in Kapitel 3.4 aufgeführten Schwächen, da sie auf dem ursprünglichen Konzept des RIP-MTI-Algorithmus basiert. Ein Teil dieser Schwächen des Konzepts konnte erst nach und nach durch die Analyse des Verhaltens des RIP-MTI-Daemons mit Hilfe des XTPeers erkannt werden. Den Quagga RIP-Daemon als Basis für die Implementierung zu nutzen hat den großen Vorteil, dass nur der wesentliche Teil des RIP-MTI-Algorithmus implementiert und gepflegt

7 http://www.quagga.net
8 Simple Network Management Protocol

werden muss. Außerdem kann bereits so demonstriert werden, dass auch eine bestehende Implementierung eines RIP-Routers um den RIP-MTI-Algorithmus erweitert werden kann.
Der Quagga RIP-MTI-Daemon enthält, neben der Erweiterung um den RIP-MTI-Algorithmus, auch eine Schnittstelle an das Analyse- und Steuerungswerkzeug XTPeer. Dabei wurde der RIP-Daemon um den XT-Server, der Steuerungsbefehle vom XTPeer entgegen nimmt, und den SL-Client, der Status-Informationen an den XTPeer sendet, erweitert. Im folgenden sind sämtliche Dateien des Quagga RIP-MTI-Daemons aufgelistet.

```
R1:/usr/local/src/quagga/ripd# ls -l
total 566
-rw-r--r--  1 quagga quagga  39442 Oct 10 22:41 ChangeLog
-rw-r--r--  1 quagga quagga  17745 Mar  3 22:31 Makefile
-rw-r--r--  1 quagga quagga    695 Oct 10 22:41 Makefile.am
-rw-r--r--  1 quagga quagga  18628 Oct 10 22:41 Makefile.in
-rw-r--r--  1 quagga quagga  16715 Oct 10 22:41 RIPv2-MIB.txt
-rw-r--r--  1 quagga quagga   7362 Oct 10 22:41 rip_debug.c
-rw-r--r--  1 quagga quagga   1817 Oct 10 22:41 rip_debug.h
-rw-r--r--  1 quagga quagga  52693 Oct 10 22:41 rip_interface.c
-rw-r--r--  1 quagga quagga   1316 Oct 10 22:41 rip_interface.h
-rw-r--r--  1 quagga quagga   7528 Dec  6 08:52 rip_main.c
-rw-r--r--  1 quagga quagga  83325 Mar  3 22:26 rip_mti.c
-rw-r--r--  1 quagga quagga  10250 Mar  3 17:10 rip_mti.h
-rw-r--r--  1 quagga quagga  11015 Oct 10 22:41 rip_offset.c
-rw-r--r--  1 quagga quagga   4792 Oct 10 22:41 rip_peer.c
-rw-r--r--  1 quagga quagga  28487 Oct 10 22:41 rip_routemap.c
-rw-r--r--  1 quagga quagga  14350 Oct 10 22:41 rip_snmp.c
-rw-r--r--  1 quagga quagga  53940 Mar  2 14:44 rip_xt.c
-rw-r--r--  1 quagga quagga   7587 Mar  2 14:04 rip_xt.h
-rw-r--r--  1 quagga quagga  18977 Oct 10 22:41 rip_zebra.c
-rw-r--r--  1 quagga quagga 133662 Mar  2 10:41 ripd.c
-rw-r--r--  1 quagga quagga  13642 Mar  2 08:01 ripd.h
-rw-r--r--  1 quagga quagga   7729 Nov 19 22:37 sl_client.c
-rw-r--r--  1 quagga quagga   2899 Jan 24 19:37 sl_client.h
```

Abbildung 4.2: Die Dateien des Quagga RIP-MTI Daemons

Die Dateien ripd.c und ripd.h gehören zum originalen Quagga RIP-Daemon. Sie beinhalten die Kern-Funktionen des RIP-Protokolls, darunter die Funktionen des RIP-Routing-Entscheidungsprozesses. Die Kern-Funktionen des RIP-MTI-Algorithmus sind in den Dateien rip_mti.c und rip_mti.h implementiert. Der XT-Server, zum Ansteuern des RIP-Daemons, ist in den Dateien rip_xt.c und rip_xt.h implementiert und der SL-Client, zum Auslesen der internen Status-Informationen, in den Dateien sl_client.c und sl_client.h.

4.2 Virtual-Network-User-Mode-Linux (VNUML)

Virtual-Network-User-Mode-Linux (VNUML) ist eine Open-Source Virtualisierungssoftware, bzw. eine Netzwerksimulationssoftware, für Linux, mit der virtuelle Linux-Rechnernetze aufgebaut werden können. Mittels VNUML kann der Einsatz beliebiger Linux-Software in reellen Netzwerken simuliert werden, um Konfigurationseinstellungen auszuarbeiten und um Funktion und Verhalten von Netzwerkprogrammen und -protokollen zu beobachten. In dieser Arbeit wurden virtuelle Linux-Rechnernetze unterschiedlichster Topologie mittels VNUML aufgebaut, um die Funktion und die Korrektheit des Quagga RIP-MTI-Daemons zu testen.

VNUML entstand im Jahr 2002 am Telematics Engineering Department der Technischen Universität Madrid, im Rahmen des Euro6IX-Projektes zur Einführung des IPv6 Standards in Europa und wird seitdem kontinuierlich weiterentwickelt. Die Software ist in Perl geschrieben und basiert auf der Virtualisierungssoftware User-Mode-Linux[9] (UML), einer Portierung des Linux-Kernels auf sich selbst, statt auf einen Prozessortyp, wie bspw. x86 oder PowerPC [Dik06]. Bei UML handelt es sich im wesentlichen um einen speziell compilierten Linux-Kernel der als gewöhnlicher Anwendungsprozess in einem Linux-System gestartet werden kann und unter dem weitere beliebige Linux-Programme ausgeführt werden können, die somit in einem virtuellen UML-Rechner laufen, abgeschottet vom System des „Host"-Rechners. Ein UML-Rechner besteht aus einem UML-Kernel und einem UML-Root-Filesystem, in das die Software installiert wird. Auf der Homepage des VNUML-Projektes[10] sind alle benötigten Teile eines UML-Rechners zum Download bereitgestellt. In dieser Arbeit wurde der UML-Kernel „linux-2.6.10-1m" und das Root-Filesystem „root_fs_tutorial-0.2.3" verwendet. VNUML wurde in der Version 1.5 verwendet.
Anhand des zuvor aufgesetzten Szenarios startet VNUML die benötigten UML-Rechner und verbindet sie zur gewünschten Netzwerk-Topologie. Zusätzlich ermöglicht VNUML eine zentrale Kontrolle der am Szenario beteiligten UML-Rechner und erlaubt so zum Beispiel das Starten und Stoppen einzelner Programme, in den UML-Rechnern, über im Szenario definierte Kommandos. Ein VNUML Szenario wird über die spezielle, auf XML[11]-basierende, VNUML-Beschreibungssprache in einer XML-Datei beschrieben. Dabei werden unter anderem die beteiligten UML-Rechner (virtuellen Maschinen), deren Eigenschaften, wie Interfaces und IP-Adressen, die Netzverbindungen zwischen den UML-Rechnern, Kommandos zur Ausführung von Programmen, sowie globale Rahmenbedingen des Szenarios beschrieben. Die Szenario-Datei wird dem VNUML-Parser übergeben, der das virtuelle Netzwerk nach der aufgeführten Szenario-Beschreibung aufbaut. Somit lassen sich schnell und mit relativ geringem Aufwand auch komplexere Linux-Rechnernetze unterschiedlichster Topologie aufbauen, in denen beliebige Linux-Programme ausgeführt und beobachtet werden können. Aus softwaretechnischer Sicht stellt VNUML eine sehr praxisnahe Umgebung zum Analysieren, Entwickeln und Testen von Netzwerkanwendungen unter Linux dar.
Um einen problemlosen Ablauf mittels VNUML zu gewährleisten sollte in einem Netzwerk-Szenario auf die folgende Konfigurationen geachtet werden.

- Im Bereich <global> muss das Tag Host-Mapping (<host_mapping>) aufgeführt sein. Dadurch werden die Hostnamen der beteiligten virtuellen Maschinen (bzw. RIP-Daemons) in die Datei „/etc/host" eingetragen, wodurch der XTPeer die IP-Adressen der virtuellen Maschinen in den zugehörigen Hostnamen umsetzen kann.
- Als Hostnamen sollten für die virtuellen Maschinen möglichst kurze Bezeichner verwendet werden, zum Beispiel R1, R2, R3, usw. (<vm name="R1">).
- Das VNUML-Management-Netzwerk muss aktiviert sein und es muss eine Managment-Netzwerk-Verbindung zwischen dem Host-Rechner und jedem beteiligten UML-Rechner existieren. Die IP-Adressen des Management-Netzwerks sollten aus dem IP-Adress-Bereich 192.168.0.0/8 stammen.
- Das Interface (eth0) <if id=0 ...> darf nicht für das virtuelle Rechnernetz vergeben werden, sondern muss als Interface für das Management-Netzwerk eingebunden werden, was VNUML in Version 1.5 automatisch macht.

9 User-Mode-Linux, http://user-mode-linux.sourceforge.net
10 Virtual User-Mode-Linux, http://www.dit.upm.es/~vnuml
11 eXtended Markup Language

- Für die im virtuellen Linux-Rechnernetz liegenden Subnetze dürfen keine IP-Adressen aus dem IP-Bereich 192.168.0.0/8 verwendet werden. In VNUML Version 1.5 ist dieses Netz dem VNUML-Management-Netzwerk vorbehalten und dient der Kontrolle der virtuellen Maschinen. IP-Adressen aus diesem IP-Bereich werden vom XTPeer generell herausgefiltert und bei der Analyse der Routing-Informationen nicht beachtet.

Auch wenn der XTPeer mittlerweile stark an VNUML und die damit aufgebauten virtuellen Linux-Rechnernetze angepasst wurde, ist seine Funktion und die Funktion des Quagga RIP-Daemons in keinster Weise von VNUML abhängig. Der XTPeer und die RIP-Daemons können auch in reellen Rechnernetzen zum Einsatz kommen.

4.3 Der XTPeer

Der XTPeer[12] ((RIP) eXternally Triggered Peer) ist ein Programm zur externen Steuerung und Analyse des Quagga RIP-Daemons. Er ist in der Programmiersprache Java implementiert und entstand in seiner jetzigen Form über verschiedene Diplomarbeiten unterschiedlicher Personen in der AG Rechnernetze an der Universität Koblenz. Die erste Version des XTPeers ist das Ergebnis der Diplomarbeit [Päh06] von Daniel Pähler, der das Basis-Programm mit der XT-Client/Server Struktur und der Möglichkeit der zentralen Steuerung der Quagga RIP-Daemons implementierte. Stefan Lange erweiterte den XTPeer in seiner Diplomarbeit [Lan07] zu einem zentralen Protokollierungs-Server, der die anfallenden Informationen über den Status der einzelnen Routen in den RIP-Daemons aufzeichnet und mit Hilfe der SL-Client/Server Struktur zum XTPeer sendet und dort zentral anzeigt. Tim Keupen erweiterte den XTPeer in seiner Diplomarbeit [Keu07] um die Möglichkeit des automatischen Generierens von Testfällen, um zum Beispiel das CTI-Problem kontrolliert zu Erzeugen und mehrmals Hintereinander ablaufen lassen zu können. In dieser Diplomarbeit wurde der XTPeer um den Umgang mit den neu im RIP-MTI-Daemon hinzugekommenen RIP-MTI-Funktionen erweitert und um die Möglichkeit, speziell den RIP-MTI-Algorithmus von außen, über die XT-Client/Server-Struktur zu steuern und die im RIP-MTI-Daemon vorhandenen Informationen über die SL-Client/Server-Struktur dem XTPeer zuzusenden und dort anzuzeigen. Zur übersichtlichen Darstellung der Informationen im XTPeer wurden auch verschiedene grafische Oberflächen implementiert.
Der XTPeer stellt zusammen mit einer VNUML-Netzwerksimulation eine Netzwerk-Testumgebung dar, die lokal auf einem Host-Rechner ausgeführt werden kann. Der XTPeer läuft dabei direkt auf dem Host-Rechner, auf dem auch mittels VNUML das virtuelle Linux-Rechnernetz gestartet wird. In den einzelnen beteiligten UML-Rechnern der VNUML-Simulation ist der Quagga RIP-Daemon gestartet, der Aufgaben des Routings für das aufgebaute VNUML-Netzwerk übernimmt. Nach dem die einzelnen Quagga RIP-Daemons gestartet wurden, können sie mit dem XTPeer verbunden werden, um von dort aus angesteuert zu werden und die Informationen ausgelesen zu bekommen. Dazu wird dem XTPeer die Management-IP-Adresse des UML-Rechners, in dem der RIP-Daemon läuft, übergeben. Ist in allen UML-Rechnern der VNUML-Simulation ein RIP-Daemon gestartet, dann kann dem XTPeer auch die dazugehörige VNUML-Szenario-Datei übergeben werden, um automatisch alle UML-Rechner einzubinden.
Wie zuvor bereits erwähnt, lässt sich der XTPeer in den XT-Client und den SL-Server unterteilen, wobei die jeweiligen Gegenparts, der XT-Server und der SL-Client, im Quagga RIP-Daemon implementiert sind. Der XT-Client des XTPeers sendet Kontrollkommandos an den XT-Server des Quagga RIP-Daemons, was die Kontrolle des Daemons von außen erlaubt, um

12 In [Päh06] RIP-XT bezeichnet

zum Beispiel das Senden von Routing-Updates zu veranlassen oder zu unterbinden. Der SL-Client des Quagga RIP-Daemons sammelt kontinuierlich Informationen über den internen Status und den Status einzelner Routen und sendet diese an den SL-Server des XTPeers, wo die Informationen dann grafisch angezeigt und ausgewertet werden. In Abbildung 4.3 ist eine schematische Darstellung der Testumgebung aus VNUML-Netzwerk und XTPeer abgebildet. Natürlich könnten die Quagga RIP-Daemons auch auf realen Rechnern eines realen Netzwerks installiert werden und über den XTPeer angesteuert werden. Hierbei fehlt allerdings die Flexibilität beim Aufbau der Netzwerk-Topologie, die ein virtuelles, mit VNUML aufgebautes Netzwerk bietet.

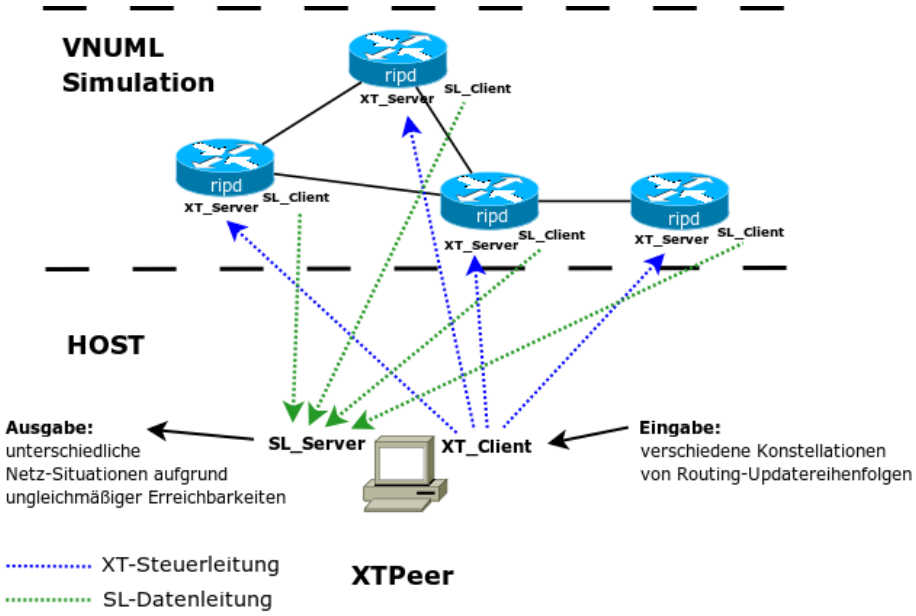


Abbildung 4.3: Schematische Darstellung der XTPeer XT-/SL-Verbindungen

Die grafische Oberfläche des XTPeers ist in zwei Abschnitte untergliedert. Im oberen Abschnitt ist die Topologie des Rechnernetzes der Testumgebung dargestellt. Zu sehen sind hier die einzelnen Rechner, deren RIP-Daemon mit dem XTPeer verbunden ist. Über den dargestellten Rechner kann der entsprechende RIP-Daemon ausgewählt und konfiguriert werden. Über einen „Klick mit der rechten Maustaste“ auf einen der abgebildeten Rechner kann zum Beispiel das in dieser Arbeit implementierte RIP-MTI-Konfigurations-Fenster geöffnet werden, um den RIP-MTI-Algorithmus des entsprechenden Daemons zu konfigurieren. Die grafische Oberfläche der RIP-MTI-Konfiguration wird in Kapitel 4.3.2 erläutert. Innerhalb der grafischen Darstellung der Rechner sind deren Interfaces angezeigt, über die das Senden von Routing-Updates direkt gesteuert werden kann. Ein „Klick mit der Maus“ auf die Abbildung des Interface veranlasst den dazugehörigen RIP-Daemon ein Routing-Update über das entsprechende Interface zu senden. Desweiteren sind hier die Subnetze des Netzwerks dargestellt. Ein „Klick mit der rechten Maustaste auf das Subnetz“ öffnet den zugehörigen Metrikgraphen, der in dieser Arbeit implementiert wurde und in Kapitel 4.3.3 erläutert wird.
Der untere Abschnitt der grafischen Oberfläche ist wiederum in zwei Bereiche unterteilt, die die internen Routing-Informationen, die der XTPeer von den einzelnen RIP-Daemons empfangen hat, grafisch präsentieren. Zum einen ist hier der Verlauf der Veränderungen der Routen in Tabellenform protokolliert. Für jeden RIP-Daemon existiert hier ein eigener Karteireiter und für jedes Subnetz, für das dieser RIP-Daemon eine Route besitzt, existiert wiederum ein weiterer Karteireiter unter dem die entsprechende Tabelle mit dem Verlauf der Veränderungen der Route

zu finden ist. Jede Zeile in der Tabelle steht dabei letztlich für eine Überprüfung der in der Routing-Tabelle des entsprechenden RIP-Daemons enthaltenen Route. Ein „Klick“ auf eine Spalte der Tabelle öffnet eine die Debug-Ausgabe, die in dieser Arbeit implementiert wurde und in Kapitel 4.3.2 erläutert wird.
Im unteren Bereich ist ein Verlaufsgraph zu finden, der den zeitlichen Verlauf einer Simulation und ein darin aufgetretenes CTI-Problem grafisch anzeigt. Für jede Route eines jeden RIP-Daemons existiert ein Verlaufsgraph, der wie die dazugehörige Tabelle, über die entsprechenden Karteireiter zu erreichen ist.
Weitere Informationen zum oberen Abschnitt der grafischen Oberfläche findet sich in [Päh06] und zum unteren Abschnitt in [Lan07].

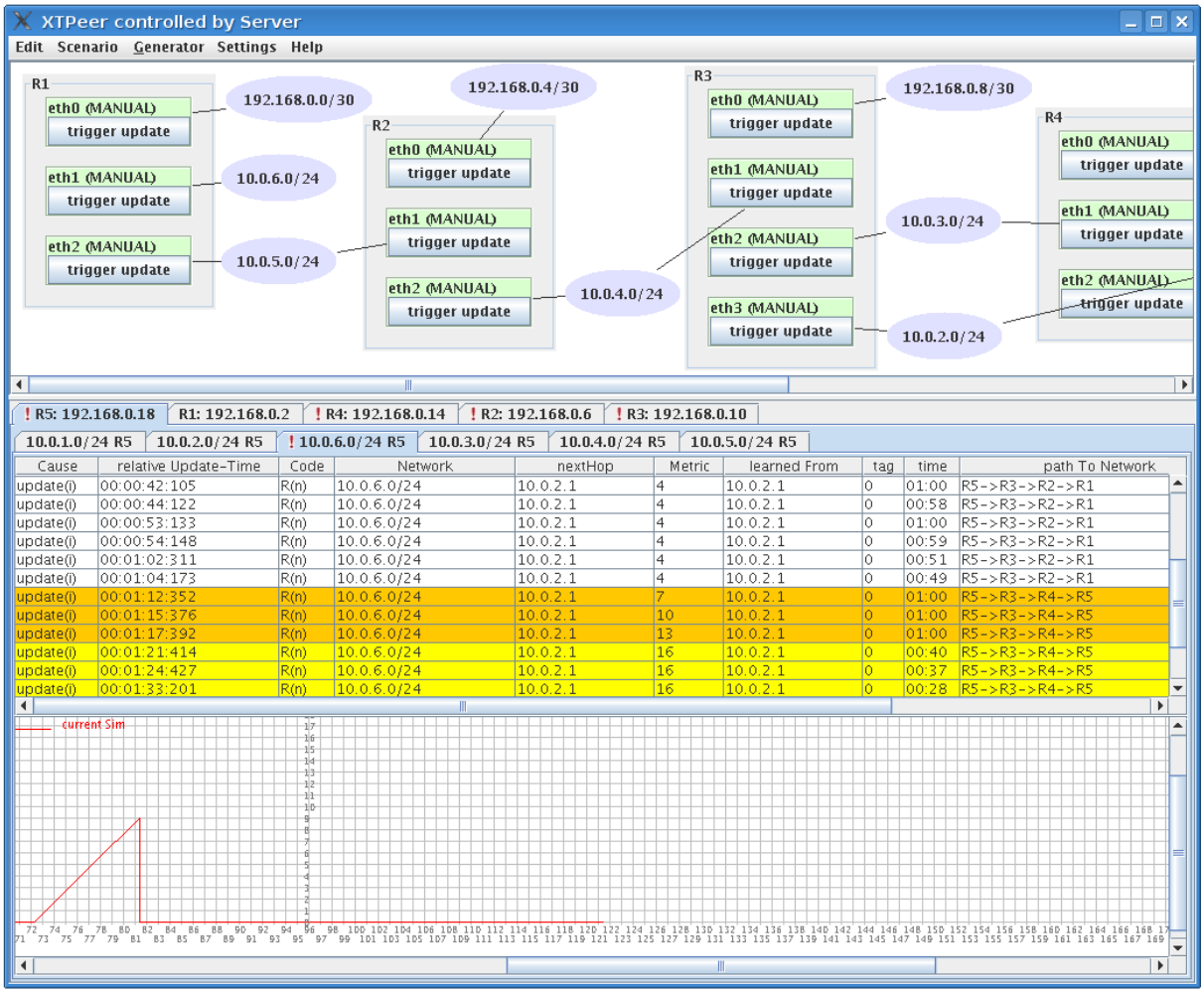


Abbildung 4.4: Bildschirmphoto des Hauptprogrammfensters des XTPeer

In Abbildung 4.4 ist in dem XTPeer das CTI-Problem aus Kapitel 2.2 protokolliert. Die verwendete Topologie ist in Abbildung 2.2 zu sehen.

4.3.1 Konfiguration des RIP-MTI-Algorithmus

Die in den XTPeer eingebundenen RIP-MTI-Daemons können vom XTPeer aus zur Laufzeit jederzeit konfiguriert werden. Die Konfigurationseinstellungen werden mittels des Klartextprotokolls der XT-Client/-Server Verbindung an die ausgewählten RIP-MTI-Daemons übertragen. Die verschiedenen Konfigurationsmöglichkeiten des RIP-MTI-Algorithmus sind in Sektionen unterteilt, die unabhängig voneinander konfiguriert werden können. Jede dieser Sektionen umfasst einen eigenständigen Bereich der Einstellungsmöglichkeiten des RIP-MTI-Daemons, der durch ein eigenes Protokollwort angesprochen wird. Die RIP-Daemons können via Telnet, standardmäßig über Port 5000, auch ohne XTPeer konfiguriert werden. Der Aufbau der Protokoll-Syntax ist im Anhang zu finden.

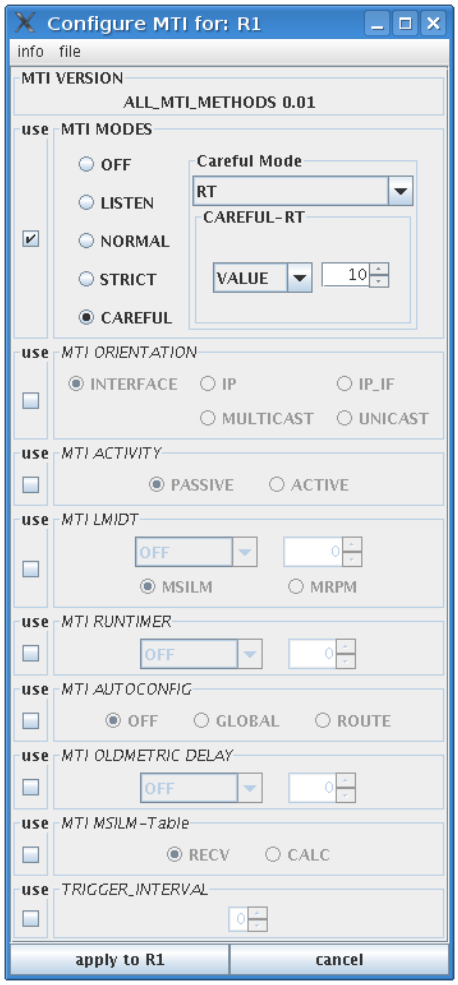


Abbildung 4.5: Konfiguration des MTI

Der XTPeer bietet zur Konfiguration des RIP-MTI-Algorithmus eine grafische Oberfläche, die in Abbildung 4.5 dargestellt ist. Die hier abgebildete Oberfläche zeigt die Konfiguration des RIP-MTI-Daemons R1. Es kann ein einzelner RIP-MTI-Daemon konfiguriert werden oder aber alle RIP-MTI-Daemons des Netzwerks zusammen, sofern denn alle gleich konfiguriert sein sollen. Wird das Konfigurations-Fenster eines einzelnen RIP-MTI-Daemons geöffnet, so wird dessen gegenwärtige Konfiguration ausgelesen und die Werte direkt im Konfigurations-Fenster als Voreinstellungen angezeigt.
Über den Menüpunkt „info“ kann zusätzlich die gegenwärtige Konfiguration einer Sektion explizit vom RIP-MTI-Daemon abgefragt werden. Über den Menüpunkt „file“ kann eine komplette Konfigurationseinstellung in einer Datei (.mticonfig) gespeichert und auch wieder aus dieser Datei geladen werden. Um eine Sektion zu konfigurieren muss zuvor die USE-Checkbox aktiviert sein. Es werden nur die Sektionen zur Konfiguration an den RIP-MTI-Daemon übertragen, bei denen die USE-Checkbox aktiviert ist. Sämtliche Veränderungen die eingestellt werden, übernimmt hier nur der angesprochene RIP-MTI-Daemon in der Topologie. Über den Menüpunkt „Scenario“ des XTPeers existiert die Möglichkeit alle RIP-MTI-Daemons des Netzwerks gleichzeitig zu konfigurieren, falls in allen Daemons der RIP-MTI-Algorithmus mit den gleichen Einstellung aktiv sein soll. Die grafische Oberfläche hierfür entspricht der für die Konfiguration eines einzelnen RIP-MTI-Daemons, allerdings sind hier keine Voreinstellungen zu finden. Soll nur eine bestimmte Sektion in allen RIP-MTI-Daemons gleich konfiguriert werden, dann darf auch nur bei dieser Sektion die USE-Checkbox aktiviert sein.
Bei der Konfiguration für alle RIP-MTI-Daemons eines Netzwerks besteht die Möglichkeit über dem Menüpunkt „info“ die Konfigurationseinstellungen aller

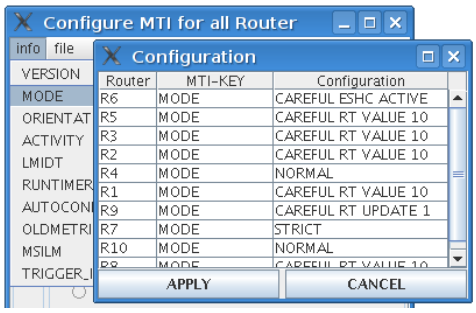


Abbildung 4.6: Das Info Menü

in den XTPeer eingebundener RIP-MTI-Daemons auszulesen, zu vergleichen und als neue Konfiguration für alle wiederum zu übernehmen. In Abbildung 4.6 sind die RIP-MTI-Mode Einstellungen aller RIP-MTI-Daemons des Netzwerks angezeigt. In der ersten Spalte der Tabelle ist der Host-Name des Daemons aufgeführt, es folgt das Protokollwort der Sektion und schließlich die aktuell eingestellte Konfiguration der einzelnen RIP-MTI-Daemons.

Für verschiedene Funktionen des RIP-MTI-Algorithmus können über die Konfiguration Zeitwerte für einen Timer eingestellt werden. Dieser Timer regelt eine bestimmte Funktionalität des RIP-MTI-Algorithmus über das eingestellte Zeitintervall. Die einstellbaren Zeitwerte bestehen dabei aus einem Protokollwort und einer natürlichen Zahl. Abhängig vom Protokollwort und der angegebenen Zahl, berechnet sich daraus die länge des Zeitintervall:

- OFF: Der Timer ist abgeschaltet, die angegebene Zahl ist Unerheblich.
- VALUE: Die angegebene Zahl wird als fester Zeitwert (in Sekunden) übernommen. Soll die zugehörige Funktion zum Beispiel für 10 Sekunden aktiviert sein, so muss die Zahl 10 angegeben werden.
- UPDATE: Die angegebene Zahl wird mit der eingestellten Zeit des Update-Timers des RIP-Daemons multipliziert. Der Update-Timer ist die Zeit zwischen dem Senden zweier Routing-Updates und beträgt standardmäßig 30 Sekunden. Soll die entsprechende RIP-MTI-Funktion für die doppelte Zeit aktiviert sein, also für die Dauer von 2 Update-Intervallen und insgesamt somit 60 Sekunden, so muss die Zahl 2 angegeben werden. Das Update-Intervall kann über die Quagga Konfiguration verändert werden.
- MAXMSILM: Die angegebene Zahl wird mit der Metrik des größten, erkannten minimalen Simple-Loops des Interfaces der alternativen Route aus der MSILM-Tabelle multipliziert. Die Funktionalität wird dann solange aufrechterhalten bis die Routing-Informationen den Simple-Loop sicher passiert haben sollten und zum Beispiel ein vorhandener Source-Loop durch die ausgesendete Information komplett überschrieben sein müsste. Dieser Wert steht im Zusammenhang mit der Frage, wie lange es durchschnittlich dauert bis eine vom Router ausgesendete Routing-Information einen Simple-Loop durchlaufen hat und wieder beim Router ankommt. Dies hängt wiederum auch mit der „Worst-Case-Time“ des CTI-Problems zusammen, wie lange es dauert bis alle am CTI-Problem beteiligten Router die Unerreichbarkeit des Subnetzes erkannt haben. In [Keu07] S.15 ff. wird dies näher erläutert. Dort wird von einer Zeit von maximal 5 Sekunden pro Router ausgegangen, bis eine Information zum nächsten Router weitergesendet wird, was wiederum von der Wartezeit zwischen zwei Triggered-Updates abgeleitet ist.
- MRPM: Diese Einstellung entspricht von der Idee her der MAXMSILM-Einstellung, baut allerdings auf der MRPM-Tabelle auf. Die angegebene Zahl wird mit dem MRPM-Wert des Interface der alternativen Route multipliziert, also dem kleinsten, erkannten minimalen Simple-Loop des Interface der alternativen Route.

Im folgenden werden die einzelnen Sektionen des Konfiguration-Fensters zum RIP-MTI-Algorithmus erläutert. Die Konfiguration über die XT-Client/Server-Verbindung beginnt für den RIP-MTI-Algorithmus immer mit dem Protokollwort „MTI“, anschließend folgt das Protokollwort der Sektion und danach entsprechend weitere Protokollwörter, die für die Übertragung der vollständigen Konfiguration benötigt werden. Die einzelnen Protokollwörter sind im folgenden in Großbuchstaben aufgelistet. Über die grafische Oberfläche im XTPeer wird das zur Konfiguration benötigte komplette Kommando aus den einzelnen Protkollwörtern generiert, so dass der Benutzer damit letztlich nicht konfrontiert wird.

- VERSION: Das Versionsfeld beinhaltet eine einfache Zeichenkette, fest implementiert im RIP-MTI-Daemon, die es ermöglichen soll, verschiedene RIP-MTI-Implementierungen mit dem gleichen XTPeer und somit während einer Simulation im gleichen Netzwerk bedienen zu können.
- MODES: Hierbei handelt es sich um die verschiedenen RIP-MTI-Modes, wie sie in Kapitel 3 erläutert werden. Dies konfiguriert die Kernfunktionalität des RIP-MTI-Algorithmus
  - OFF: Schaltet den RIP-MTI-Algorithmus im Quagga RIP-Daemon komplett aus. Es handelt sich hierbei um einen globalen Ausschalter. Die Implementierung des RIP-MTI-Algorithmus wird nicht mehr betreten, womit auch alle anderen Einstellungen der RIP-MTI-Konfiguration unwirksam sind. Es arbeitet sozusagen der konventionelle Quagga RIP-Daemon.
  - LISTEN: Hier ist der RIP-MTI-Algorithmus zwar aktiv, hat aber keinen Einfluss auf den Routing-Prozess und verhindert ein auftretendes CTI-Problem auch nicht. Es werden lediglich die Daten-Strukturen des RIP-MTI-Algorithmus, wie MSILM- und MRPM-Tabelle, mit Informationen gefüllt, die dann auch im XTPeer angezeigt werden. Diese Einstellung ist zum Beispiel dazu geeignet, die Informationen des RIP-MTI-Algorithmus während das CTI-Problem abläuft zu sammeln und zu analysieren. Desweiteren können auch einige zusätzliche Funktionen, wie der LMIDT (siehe Kapitel 5.6), mit deaktivierten RIP-MTI-Modes getestet werden.
  - NORMAL: Aktiviert den RIP-MTI-Normal-Mode. Dies entspricht dem originalen RIP-MTI-Algorithmus wie er von A. Schmid in [Sch99] entworfen wurde und in Kapitel 3.3 vorgestellt wird.
  - STRICT: Aktiviert den, in Kaptel 3.4.1 beschrieben, RIP-MTI-Strict-Mode. Dies entspricht der Erweiterung des RIP-MTI-Algorithmus von T. Kleemann aus [Kle01], um dem CTI-Problem auch in stark vermaschten Topologien zu begegnen.
  - CAREFUL: Aktiviert eine der in Kapitel 3.4.1 vorgestellten Varianten des RIP-MTI-Careful-Mode. Zusätzlich zu dieser Grundeinstellung muss noch die entsprechende Variante aktiviert werden. Der RIP-MTI-Careful-Mode basiert auf dem RIP-MTI-Strict-Mode, erweitert diesen aber um weitere Kriterien, die sich je nach Variante darin unterscheiden, eine gültige, alternative Route, die den X-/Y-Test nicht bestanden hat und abgelehnt wurde, schnellstmöglich doch noch aufzunehmen.
    - DT (Deny Timer): Aktiviert den RIP-MTI-Careful-DT-Mode. Hier können einige der oben erläuterten Zeitwerte für den Timer eingestellt werden. Nach Ablauf des Zeitintervalls wird eine zurückgehaltene und zwischengespeicherte Route als gültig anerkannt und in die Routing-Tabelle eingetragen.
    - RT (Request Timer): Aktiviert den RIP-MTI-Careful-RT-Mode. Nach Ablauf des eingestellten Zeitintervalls wird eine zurückgehaltene Route als gültig anerkannt und über eine Request-Nachricht von den benachbarten Routern abgefragt. Der RIP-MTI-Careful-RT-Mode wurde so implementiert, dass er die MSILM-Tabelle nicht benötigt, weswegen hier nicht die gleichen Zeiteinstellungen zu finden sind, wie beim RIP-MTI-Careful-DT-Mode. Vom Prinzip her sind sich diese beiden RIP-MTI-Modes aber sehr ähnlich.
    - ESHC (External Split Horizon Check): Aktiviert den RIP-MTI-Careful-ESHC-Mode. Hier muss zusätzlich die Aktivität eingestellt werden. Beim aktiven Mode werden zusätzliche Response- und Request-Nachrichten gesendet, um die Liste der

Interfaces, über die die Existenz einer alternativen Route geprüft wird, schneller zu überprüfen und so die Konvergenz zu beschleunigen. Beim passiven Mode werden keine zusätzlichen RIP-Nachrichten gesendet.

- ORIENTATION: Hier wird die Orientierung des RIP-MTI-Algorithmus eingestellt. Zur Wahl steht die Orientierung über Interfaces (INTERFACE), über die Learned-From-IP-Adresse (IP) der eingetroffenen Routing-Updates (siehe Kapitel 3.4.4) und beides in Kombination. Die Einstellungen MULTICAST und UNICAST bestimmen, wie die aktiven RIP-MTI-Modes RIP-Nachrichten versenden, über das entsprechende Interface via Multicast oder an die IP-Adresse des benachbarten Routers via Unicast. Bei einer Orientierung über Interfaces können RIP-Nachrichten nur via Multicast gesendet werden.
- ACTIVITY: Eine zusätzliche Möglichkeit für den RIP-MTI-Algorithmus bei Ablehnung einer alternativen Route eine Response-Nachricht auszusenden um die Konvergenz des Netzwerks zu Beschleunigen. Vor allem für die eigentlich passiv entworfenen Modes, wie den RIP-MTI-Normal-Mode und den RIP-MTI-Strict-Mode ist diese Einstellung interessant, da der RIP-MTI-Daemon somit auch die Routing-Information der benachbarten Router in einer Netzwerkschleife löschen kann, sofern ein Source-Loop vorliegt. Die RIP-MTI-Careful-Mode Varianten senden, unabhängig von dieser Einstellung, ohnehin eine Response-Nachricht bei Ablehnung einer Route aus. Über die Einstellung ACTIVE wird nach Ablehnung einer Route eine zusätzliche Response-Nachricht an die benachbarten Router gesendet. Über die Einstellung PASSIVE wird keine weitere Nachricht gesendet.
- LMIDT (Loop-Metric-Increase-Deny-Timer): Aktiviert den LMIDT. Der LMIDT ist kein „vollwertiger“ RIP-MTI-Mode. Er lehnt ein Routing-Update mit einer höheren Metrik für eine Route vom gleichen Router ab, wenn die Summe der Metrik der vorhandenen Route in der Routing-Tabelle und die Metrik des zugehörigen Simple-Loops gleich ist mit der Metrik der neuen Route. Der LMIDT kann unabhängig von den RIP-MTI-Modes im RIP-MTI-Daemon aktiv sein, wenn in der Sektion Mode LISTEN eingestellt wurde. Der LMIDT kann aber auch zusammen mit einem RIP-MTI-Mode funktionieren, quasi als zweite zu überwindende Barriere für das CTI-Problem. Über MRPM und MSILM wird eingestellt, ob der LMIDT die Simple-Loops für den Vergleich aus der MSILM- oder MRPM-Tabelle verwenden soll. Zusätzlich muss einer der oben erläuterten Zeitwerte eingestellt werden, nach dessen Ablauf eine zurückgehaltene und zwischengespeicherte Route als gültig anerkannt und nachträglich in die Routing-Tabelle eingetragen wird.
- RUNTIMER: Aktiviert eine generelle Laufzeitbeschränkung des RIP-MTI-Algorithmus. Diese Einstellung ist eigentlich nur für den RIP-MTI-Strict-Mode interessant, um dessen Funktionalität auf ein kürzeres Zeitintervall zu begrenzen. Der Runtimer wird mit dem ersten Ablehnen einer alternativen Route gestartet. Nach Ablauf des Runtimers wird eine alternative Route aus der gleichen Richtung nicht mehr abgelehnt, wenn sie ein weiteres mal eintrifft. Vom Konzept her entspricht der Runtimer zusammen mit dem RIP-MTI-Strict-Mode dem RIP-MTI-Careful-RT/DT-Mode, ohne das Senden einer Request-Nachricht, bzw. zwischenspeichern der abgelehnten, alternativen Route.
- AUTOCONFIG: Über diese Konfiguration hat der RIP-MTI-Algorithmus die Möglichkeit den verwendeten RIP-MTI-Mode selbst zu wählen. Die Autoconfig-Funktion ist derzeit noch recht einfach implementiert. Sie kann lediglich in Abhängigkeit der erkannten Topologie und über die Einträge in der MSILM-Tabelle zwischen dem RIP-MTI-Normal-Mode und einer voreingestellten RIP-MTI-Careful-Mode Variante wählen. Ist GLOBAL aktiviert, so überprüft der RIP-MTI-Algorithmus den eingestellten RIP-MTI-Mode immer am Anfang, beim betreten der Implementierung. Ist ROUTE aktiviert, so wird der einge-

stellte RIP-MTI-Mode immer nur kurz vor der Überprüfung einer alternativen Route überprüft, wenn die Route in der Routing-Tabelle ausgefallen ist und durch die alternative Route ersetzt werden könnte. Mit OFF ist die Autoconfig-Funktion ausgeschaltet.

- OLDMETRIC_DELAY: Diese Einstellung berücksichtigt die Korrektur aus Kapitel 3.4.2. Steht der Wert des Timers auf OFF, so wird auch der X-Test verwendet um Simple-Loops zu erkennen. Als „oldmetric“ für den RIP-MTI-Algorithmus wird hier die letzte Metrik der Route verwendet, die sie vor ihrem Ausfall hatte. Steht der Wert des Timers auf einer anderen Einstellung, so wird nur noch der Y-Test zur Erkennung von Simple-Loops verwendet. Eine kleinere Metrik in der Variable oldmetric wird erst durch eine größere Metrik einer neuen Route ersetzt, wenn nach Ablauf des eingestellten Zeitwertes erneut ein Routing-Update für die Route eintrifft. Auch das zur Metrik, bzw. zur Route, gehörende Interface wird berücksichtigt, da normalerweise Routing-Updates zu einer Route, die über das gleiche Interface wie zuvor eintreffen nur vom LMIDT überprüft werden.
- MSILM_TABLE: Dieser Punkt berücksichtigt die Korrektur aus Kapitel 3.4.3. In Netzwerken mit entsprechend komplexen Topologien können eventuell nicht alle Simple-Loops über die eintreffenden Routing-Updates gefunden werden, sehr wohl aber über eine transitive Berechnung der bereits erkannten Simple-Loops. Der Punkt CALC aktiviert diese transitive Berechnung, während der Punkt RECV sie wieder deaktiviert und die MSILM-Tabelle allein über die empfangenen Routing-Updates aufbaut.
- TRIGGER_INTERVAL: Die aktiven RIP-MTI-Modes senden bei Ablehnung einer Route eine Response- oder Request-Nachricht an die benachbarten Router, um einen eventuell vorhandenen Source-Loop in der Netzwerkschleife zu löschen oder eine fragliche Route erneut abzufragen. Um nicht für jede abgelehnte Route eine eigene RIP-Nachricht extra an die benachbarten Router zu senden, werden die RIP-Nachrichten nicht sofort generiert und gesendet. Erst nach Ablauf des eingestellten Zeitintervalls wird eine einzige RIP-Nachricht an den benachbarten Router gesendet, vertretend für alle in der Laufzeit des Trigger-Intervalls abgelehnten Routen und angefallenen RIP-Nachrichten. Dies sollte die Anzahl der gesendeten Nachrichten durch den RIP-MTI-Algorithmus insgesamt erheblich reduzieren. Zusätzlich werden beim RIP-MTI-Careful-RT-Mode Request-Nachrichten mit gezielten Subnetz-Anfragen gesendet, wenn das Trigger-Intervall auf OFF steht.

Eine Auflistung der XT-Protkoll-Syntax des RIP-MTI-Algorithmus in EBNF ist im Anhang B zu finden.

4.3.2 Die Debug-Ausgabe

Die RIP-MTI-Debug-Ausgabe ist eine grafische Ansicht aller Informationen des RIP-MTI-Algorithmus, die vom SL-Client des RIP-MTI-Daemons gesammelt und an den SL-Server des XT-Peers übermittelt wurden. Die SL-Client/Server-Verbindung wurde von S. Lange in [Lan07] entworfen und hier um die neuen Informationen entsprechend erweitert. Neben einigen Steuernachrichten, die dem Verbindungsaufbau und -abbau, sowie der Verbindungsüberwachung zwischen SL-Client und -Server dienen, existieren fünf Nachrichtentypen, über die interne Router- und Routing-Informationen der RIP-MTI-Daemons an den XTPeer übermittelt werden. In dieser Arbeit wurde die MSG_MTI_DEBUG_INFO-Nachricht entsprechend um die hinzugekommen Information des RIP-MTI-Algorithmus erweitert. Neu hinzugekommen ist die Nachricht MSG_RTE_GARBAGE, die generiert wird wenn für eine Route keine Routing-

Updates mehr eintreffen, der Garbage-Collection-Timer abgelaufen ist und die Route nun aus der Routing-Tabelle gelöscht wird. Wurde eine Route aus der Routing-Tabelle gelöscht, dann hat der RIP-MTI-Algorithmus auch keine Vergleichsmöglichkeit mehr, um eine angebotene, alternative Route zu überprüfen.

- MSG_INITIAL:
  Nach dem Start des RIP-Daemons verfügt dieser nur über Informationen zu Subnetzen, die direkt an seinen Interfaces anliegen. Sie werden mit Metrik 1 in die Routing-Tabelle eingetragen. Nach dem Herstellen der Verbindung zwischen dem XTPeer und dem RIP-Daemon werden diese initialen Routen direkt an den XTPeer übermittelt.

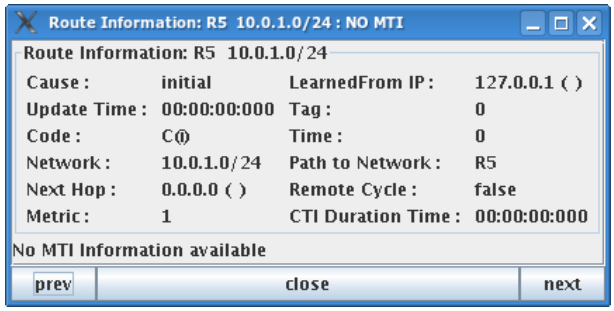


Abbildung 4.7: Initial-Nachricht

- MSG_UPDATE:
  Diese Nachrichten werden vom SL-Client erzeugt, wenn eine Route in der Routing-Tabelle des RIP-Deamons, aufgrund eines eintreffenden Routing-Updates überprüft oder verändert wird. Auch wenn die Route neu in die Routing-Tabelle eingetragen oder durch eine andere Route ersetzt oder auch nur mit ihr verglichen wird, teilt der SL-Client dies dem SL-Server über eine solche Update-Nachricht mit.

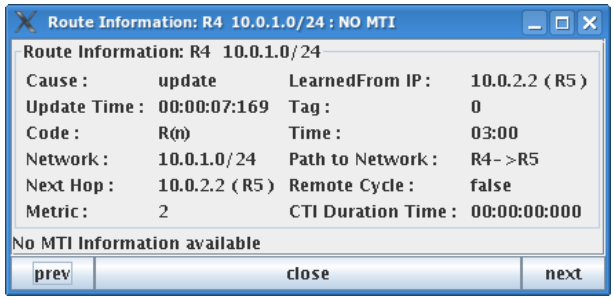


Abbildung 4.8: Update-Nachricht

- MSG_RTE_TIMEOUT:
  Wird eine Route in der Routing-Tabelle für 180 Sekunden (Standard) nicht bestätigt, dann läuft der Timeout-Timer der Route ab und der Garbage-Collection-Timer wird gestartet. Die Metrik der Route wird auf RIP-Infinity (16) gesetzt und das Subnetz, zu dem die Route führt, gilt fortan als unerreichbar. Der SL-Client des RIP-Daemons erzeugt eine Timeout-Nachricht um den SL-Server des XTPeers darüber zu informieren.

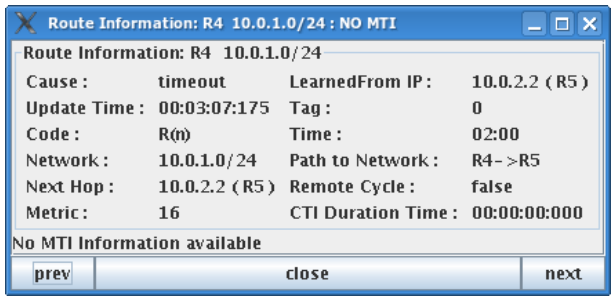


Abbildung 4.9: Timeout-Nachricht

- MSG_RTE_GARBAGE:
  Die Garbage-Nachricht wurde in dieser Arbeit neu hinzugefügt. Sie ist von der MSG_RTE_TIMEOUT abgeleitet. Der SL-Client sendet eine Garbage-Nachricht an den SL-Server des XTPeers, wenn der Garbage-Collection-Timer für eine Route abgelaufen ist und die Route nun komplett aus der Routing-Tabelle des RIP-Daemons gelöscht wird.

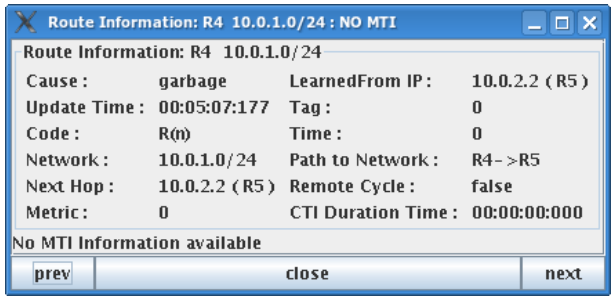


Abbildung 4.10: Garbage(-Collector)-Nachricht

• MSG_MTI_DEBUG_INFO: Der SL-Client sendet eine RIP-MTI-Nachricht an den SL-Server des XTPeers, wenn die Implementierung des RIP-MTI-Algorithmus betreten wird. Die RIP-MTI-Nachricht wird normalerweise immer mit einer Update-Nachricht erzeugt, da der RIP-MTI-Algorithmus bislang nur betreten wird, wenn ein Routing-Update eintrifft. Die Information in der Update-Nachricht ist das direkte Ergebnis der Entscheidung des RIP-MTI-Algorithmus. Da sich die Informationen ergänzen werden sie immer zusammen in einer gemeinsamen Debug-Ausgabe angezeigt.

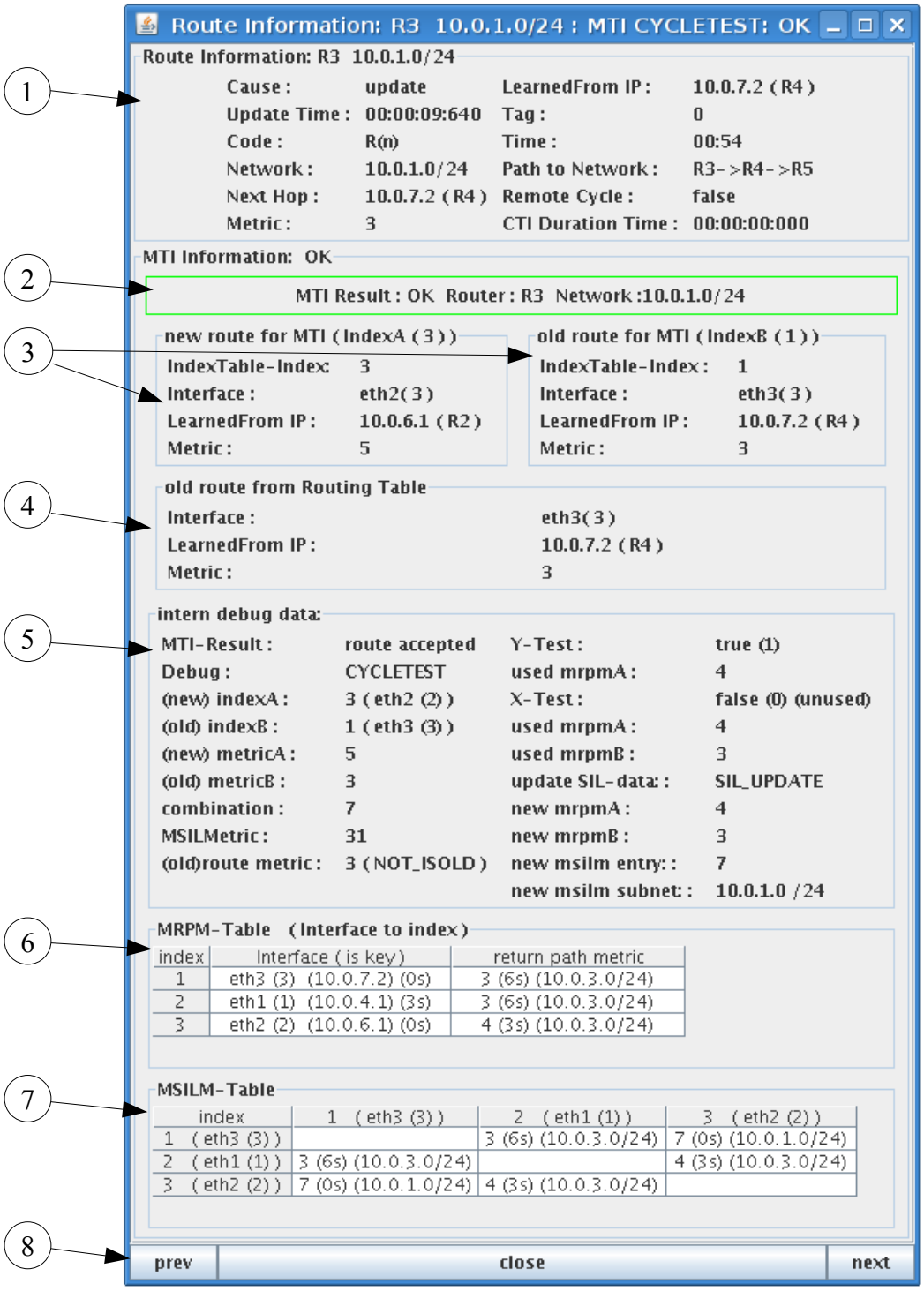


Abbildung 4.11: Update-Message mit MTI-Info-Debug-Message

Die einzelnen Bereiche der Update- und RIP-MTI-Nachricht unter den in Abbildung 4.11 aufgeführten Punkten werden im folgenden erläutert.

1) Hier befindet sich die Information einer Route, wie sie aktuell in der Routing-Tabelle des RIP-Daemons zu finden ist. Der RIP-MTI-Algorithmus wurde zuvor angewandt und die hier aufgeführte Information ist mit das Ergebnis seiner Entscheidungen. Die gleiche Information ist auch in der Verlaufs-Tabelle unter dem Karteireiter der jeweiligen Route des entsprechenden RIP-Daemons, im Hauptprogrammfenster des XTPeers, aufgeführt.

Bezeichner	**Beschreibung**
Cause	Grund für die Generierung der SL-Log-Meldung. (bezieht sich auf die oben genannten Nachrichtentypen: initial, update, timeout, garbage)
Update-Time	Zeitpunkt der Generierung der Log-Meldung
Code	Wie die Route gelernt wurde. (Bsp: R: RIP; C: Connected; n: Normal; i: Interface; ...)
Network	Das Ziel-Subnetz, zu dem die Route hinführt.
Next Hop	Der nächste Hop (Router) auf dem Weg zum Subnetz.
Metric	Die Metrik der Route (Distanz) zum Subnetz (= Anzahl Router)
LearnedFrom IP	Die IP-Adresse des Routers, von dem die Route übernommen wurde.
Tag	Dient der Unterscheidung zwischen einer intern, aus dem Autonomen System (AS), gelernten Route (IGP) und einer extern, von außerhalb des AS, gelernten Route (EGP). Wird auch zur Kennzeichnung der RIP-Nachrichten mit Authentifizierung verwendet.
Time	Der RIP Timer dieser Route. Zeigt die Zeit an, wann das letzte Routing-Update mit einer Bestätigung der Route eingetroffen ist.
Path-to-Network	Der im XTPeer berechnete Pfad der Route, bzw. der Weg des letzten Routing-Updates, durch das Netzwerk zum Ziel-Subnetz.
Remote Cycle	Anzeige, ob ein Routing-Loop vorliegt, an dem dieser RIP-Daemon zwar nicht direkt beteiligt ist, aber von dessen Auswirkungen auf die Erreichbarkeit des Subnetzes er indirekt betroffen ist.
CTI Duration Time	Dauer des CTI-Problems bis zu dieser Log-Meldung

2) Eine kleine einleitende Übersicht über das Ergebnis der Überprüfung des RIP-MTI-Algorithmus, der Hostname des Routers und das Subnetz, dessen Route hier überprüft wurde.
3) Der RIP-MTI-Algorithmus vergleicht die vorhandene Route mit einer neu angebotenen, alternativen Route zum gleichen Subnetz. Hier sind die Daten der Routen angezeigt, die miteinander im RIP-MTI-Algorithmus verglichen wurden, darunter der Index, der die Position des Eintrags in der MRPM-Tabelle anzeigt, das zugehörige Interface, die Learned-From-IP-Adresse und der zugehörige Hostname des benachbarten Routers von dem die Route übernommen wurde sowie die Metrik der Route. Wird die in Kapitel 3.4.2 erwähnte Reduzierung auf den Y-Test angewendet (oldmetric_delay), dann entspricht die vorhandene Route, die der RIP-MTI-Algorithmus zum Vergleich nimmt, nicht unbedingt der letzten gültigen Route aus der Routing-Tabelle.
4) Hier ist die zuletzt gültige Route aus der Routing-Tabelle aufgeführt. Diese Information wird aktualisiert sobald ein Routing-Update für die Route mit RIP-Infinity (16) eintrifft.

5) In diesem Abschnitt sind detailliert die Debug-Informationen der internen Berechnung des RIP-MIT-Algorithmus aufgelistet. Dabei werden eine Reihe von Schlüsselwörtern der erweiterten SL-Protokoll-Syntax angezeigt, die in der folgenden Tabelle erläutert werden.

Bezeichner	**Beschreibung**
MTI-Result : *route accepted / rejected*	Der RIP-MTI-Algorithmus akzeptiert oder verwirft diese Route. Akzeptiert er die Route, dann bedeutet das allerdings nicht unbedingt, dass die Route auch den restlichen Routing-Prozess übersteht und schließlich in die Routing-Tabelle eingetragen wird.
Debug : *RUNTIMER_ACK / CAREFUL_ACK / LMIDT(_ACK) / EQUAL_INDICES / HIGH_METRIC / CYCLETEST*	Der RIP-MTI-Algorithmus kann aus unterschiedlichen Gründen mit einem Ergebnis über die alternative Route verlassen werden, was auch abhängig von der eingestellten Konfiguration ist. Hier wird angezeigt welcher Bereich der Implementierung des RIP-MTI-Algorithmus für das erhaltene Ergebnis durchlaufen wurde. Die einzelnen Bezeichner werden im folgenden noch genauer erläutert.
RUNTIMER-ACK	Der Runtimer war aktiv und ist nun wieder beendet
runtimer	Die Variable zeigt an, vor wie vielen Sekunden der Runtimer gestartet wurde. Bei der Einstellung des Runtimer handelt es sich um eine Mindestlaufzeit. Erst die Routen die nach Ablauf des Runtimers eintreffen, umgehen den RIP-MTI-Algorithmus und können zumindest von diesem nicht mehr abgelehnt werden.
LMIDT	Der LMIDT wurde aktiv, da die Metrik der neuen Route um die Metrik eines Simple-Loops gewachsen ist, was Ergebnis eines aufgetretenen CTI-Problems sein kann.
IndexA, IndexB	Die zugehörigen IndexA- und indexB-Werte der MRPM-Tabelle
SIL-Metric	Die Metrik des Simple-Loops, die der Differenz der Metriken der vorhandenen und neuen Route entspricht.
blocking time	die vom LMIDT gesetzte Sperrzeit der Route in Sekunden
route check : *PASS*	Der LMIDT wurde nicht aktiviert. Die Route ist ok.
LMIDT-ACK	Der LMIDT ist aktiv.
route check : *METRIC_UPDATE*	Die vom LMIDT zurückgehaltene Route wurde aktualisiert
new metric new learned from IP	Die Daten der zurückgehaltenen Route, die Metrik und die LearnedFrom-IP-Adresse, wurden aktualisiert.
route check: *PASS / FAIL*	Die Route wurde angenommen, oder auch nicht
EQUAL_INDICES	Das Routing-Update zur Route trifft über das gleiche Interface (LearnedFrom-IP-Adresse) ein, wie bereits das Vorherige.
CAREFUL_ACK	Eine der RIP-MTI-Careful-Mode Varianten ist aktiv und überprüft die eingehenden Routen.
Mode : *DT / RT / ESHC*	Die aktive RIP-MTI-Careful-Mode Variante
route check :	Die Daten der vom RIP-MTI-Careful-DT zurückgehaltenen Route

METRIC_UPDATE	werden aktualisiert.
new metric new learned from IP	Die Daten der Route werden aktualisiert, neue Metrik, neue LearnedFrom-IP-Adresse
route check: *PASS/FAIL*	Die durch den RIP-MTI-Careful-Mode überprüfte Route war ok und wurde übernommen / war nicht ok und wurde abgelehnt.
ESHC-List : *1,2,...*	Die Liste der Interfaces, über die der RIP-MTI-Careful-ESHC-Mode die Existenz einer alternativen Route überprüfen wird.
HIGH_METRIC	Die Metrik der alternativen Route ist zu hoch für den X/Y-Test.
(new) route metricA (old) route metricB	Die Metrik der alternativen Route und der Route aus der Routing-Tabelle. Beide Metriken müssen kleiner RIP-Infinity sein.
CYCLETEST	Die Funktion cycletest() des RIP-MTI-Algorithmus wurde betreten. Dies ist die Kernfunktion des Algorithmus. Hier werden Simple-Loops erkannt und Source-Loops vermieden.
(new) indexA (old) indexB	Die Indizes der Einträge aus der MRPM-Tabelle des zur alternativen und zur vorhandenen Route gehörenden Interfaces
(new) metricA (old) metricB	Die Metrik der alternativen Route (über IndexA) und die Metrik der vorhandenen Route vor ihrem Ausfall (über IndexB).
combination	Metrik der Routen-Kombination über IndexA und IndexB
MSILMetric	Der MSILM-Wert zwischen indexA und IndexB
(old) route metric	Die letzte gültige Metrik, die die Route aus der Routing-Tabelle vor ihrem Ausfall hatte.
used MTI-Mode : *LISTEN / NORMAL / STRICT / CAREFUL*	Der verwendete RIP-MTI-Mode. Interessant vor allem dann, wenn die automatische Selbst-Konfiguration (Autoconfig) des RIP-MTI-Algorithmus aktiv ist.
Y-Test : *true / false* *EQUAL_METRICS*	Ergebnis des Y-Tests des RIP-MTI-Algorithmus. Wurde eine Y-Kombination gefunden oder sind die Metriken der beiden Routen nicht gleich, dann existiert kein Simple Loop.
X-Test: *true / false* *(un)used*	Ergebnis des X-Tests. Wurde eine X-Kombination gefunden, dann existiert kein Simple-Loop. Der X-Test kann an- und abgeschaltet werden. Steht die Einstellung bei OLDMETRIC_DELAY auf OFF, so ist der X-Test aktiv (used), ansonsten inaktiv (unused).
used mrpmA, used mrpmB	Vom X-/Y-Test verwendete MRPM-Werte aus der MRPM-Tabelle
X/Y-Test: *NO_XY (no SIL found)*	Ergebnis des X/Y-Tests: Keiner der Tests war erfolgreich, es konnte kein Simple-Loop gefunden werden.
DT/RT blocking Time	Die Sperrzeit des RIP-MTI-Careful-DT/RT-Mode in Sekunden
DT/RT already active	Der RIP-MTI-Careful-DT/RT-Mode wurde bereits zuvor aktiviert, womöglich durch die Überprüfung einer Route zu einem anderen Subnetz. Hier wäre er erneut aktiviert worden.
SIL check: *combination>MSILM*	Die Routen-Kombination ist ein Simple-Loop deren Metrik größer ist als der vorhandene MSILM-Wert. Er wird verworfen.

update MSIL-data: *SIL_UPDATE*	Ein vorhandener Eintrag in der MSILM-Tabelle, sowie der zugehörige MRPM-Wert wurden bestätigt oder aufgrund einer kleineren Metrik ersetzt.
new mrpmA, new mrpmB	Die Einträge in der MRPM-Tabelle der Interfaces A und B werden bestätigt oder durch kleinere Werte ersetzt.
new msilm-entry	Der Eintrag in der MSILM-Tabelle des Interfaces A und B wurde bestätigt oder durch einen kleineren Wert ersetzt
new msilm-subnet	Das Subnetz, dass den MSILM-Eintrag erzeugt hat, wurde bestätigt oder ersetzt

6) Die MRPM-Tabelle zeigt die in Kapitel 3.1 vorgestellten Minimalen-Return-Pfad-Metriken (MRPM) an. Die MRPM-Tabelle ist vom Prinzip her eine Hash-Tabelle, wobei der Schlüssel, in Abhängigkeit zur eingestellten Orientierung, entweder das Interface oder die Learned-From-IP-Adresse ist. Die erste Spalte der MRPM-Tabelle ist grau untermalt und enthält die Position des Eintrags in der Tabelle, die überall dort in der Debug-Ausgabe zu finden ist, wo „Index“(A/B) angegeben ist. In der zweiten Spalte befindet sich an erster Stelle der identifizierende Schlüssel. Ist die Learned-From-IP-Adresse der Schlüssel, so muss sie durch eingehende Routing-Updates regelmäßig bestätigt werden, was durch den angegebenen Timer (in Sekunden) angezeigt wird. In der dritten Spalte ist der eigentliche MRPM-Wert zu finden, der als Referenz auf einen Eintrag aus der MSILM-Tabelle implementiert ist, weswegen der gleiche Eintrag auch dort zu sehen ist. Der Timer hier zeigt an, wann der MRPM-Wert das letzte mal bestätigt wurde. Der aufgeführte IP-Adressbereich gehört dem Subnetz aus dem Simple-Loop, das für den MRPM-Wert „verantwortlich“ ist.

7) Die MSILM-Tabelle zeigt die in Kapitel 3.1 vorgestellten Minimalen-Simple-Loop-Metriken (MSILM) an. Der Inhalt ist gleich dem Inhalt der dritten Spalte der MRPM-Tabelle. Der MRPM-Eintrag ist eine Referenz auf den zugehörigen MSILM-Eintrag. Leere Felder in der MSILM-Tabelle sind gleichbedeutend mit Feldern, in denen ein MSILM-Wert von RIP-MTI-Infinity (31) eingetragen ist. Sie sind aber ein Zeichen dafür, dass der RIP-MTI-Algorithmus zwischen den zugehörigen Interfaces noch keinen Simple-Loop gesucht hat.

8) Die Datenstruktur, in die die SL-Log-Meldungen im XTPeer gespeichert werden, wurde von S. Lange in [Lan07] als doppelt-verkettete Liste realisiert. Die Buttons „prev“ und „next“ tragen dem Rechnung und öffnen jeweils die Debug-Ausgabe der vorherigen bzw. nachfolgenden SL-Log-Meldung der gleichen Route des gleichen RIP-Daemons.

Eine Beschreibung der, für die Debug-Ausgabe, erweiterten SL-Protokoll-Syntax in EBNF ist im Anhang C zu finden.

4.3.3 Der Metrikgraph

Der Metrikgraph ist ein, in dieser Arbeit, in den XTPeer implementierter Funktionsgraph, der den zeitlichen Verlauf der Veränderungen der Metrik der einzelnen Routen für jeden RIP-MTI-Daemon anzeigt. Vom Informationsgehalt her basiert er auf der Verlaufs-Tabelle, die im Hauptprogrammfenster des XTPeers zu finden ist. Der Metrikgraph wird über den Menüpunkt „Scenario ->show Network Metricgraph->Network“ geöffnet, oder mit einem „Klick mit der rechten Maustaste“ auf das gewünschte Subnetz in der Topologie-Ansicht des XTPeers. In dem

geöffneten Fenster ist für jeden RIP-MTI-Daemon, der über eine Route zu dem entsprechenden Subnetz verfügt, ein eigener Metrikgraph angezeigt.
Die Y-Achse des Koordinatensystems des Metrikgraphen wird durch die Metrik von 0 bis 16 unterteilt. Die 0 bedeutet dabei das der RIP-MTI-Daemon derzeit über keine Route zum Subnetz in seiner Routing-Tabelle verfügt. Die 16 entspricht RIP-Infinity und bedeutet das der RIP-MTI-Daemon bis vor kurzem eine gültige Route zum Subnetz hatte, die nun aber ausgefallen ist. Die Y-Achse wurde so implementiert, das der höchste Metrik-Wert abhängig ist vom eingestellten RIP-Infinity im Quagga RIP-MTI-Daemon.
Die X-Achse ist durch die Anzahl der vom SL-Server empfangenen SL-Log-Meldungen unterteilt. Eine SL-Log-Meldung wird im RIP-Daemon immer dann erzeugt, wenn die vorhandene Route in irgendeiner Form überprüft, neu eingefügt oder ausgetauscht wurde, sowie auch beim Ablauf des Timout- oder Garbage-Collector-Timers. Die X-Achse bietet zwei Möglichkeiten der Anzeige. Zum einen können die X-Koordinaten der Graphenpunkte mit festen Abständen dargestellt werden, was einer sehr übersichtlichen Ansicht entspricht. Zum Anderen können die Abstände der X-Koordinaten zeitlich synchronisiert, nach der Generierungszeit der SL-Log-Meldungen, angezeigt werden. Da sich die Systemzeit der UML-Rechner der VNUML-Netzwerksimulation mit der Systemzeit des Hostsystem synchronisieren, kann so, in gewissem Maße, der Verlauf einer kompletten Route über alle beteiligten RIP-Daemons zeitlich nachvollzogen und verglichen werden.
In Abbildung 4.12 zeigen die beiden Bildschirmphotos einen unsynchronisierten (links) und einen synchronisierten (rechts) Metrikgraphen des gleichen Verlaufs der Metrik einer Route in den einzelnen Routern des Netzwerks. Zu sehen ist hier der Ablauf des CTI-Problems aus dem Beispiel in Kapitel 2.2 aus Abbildung 2.3.

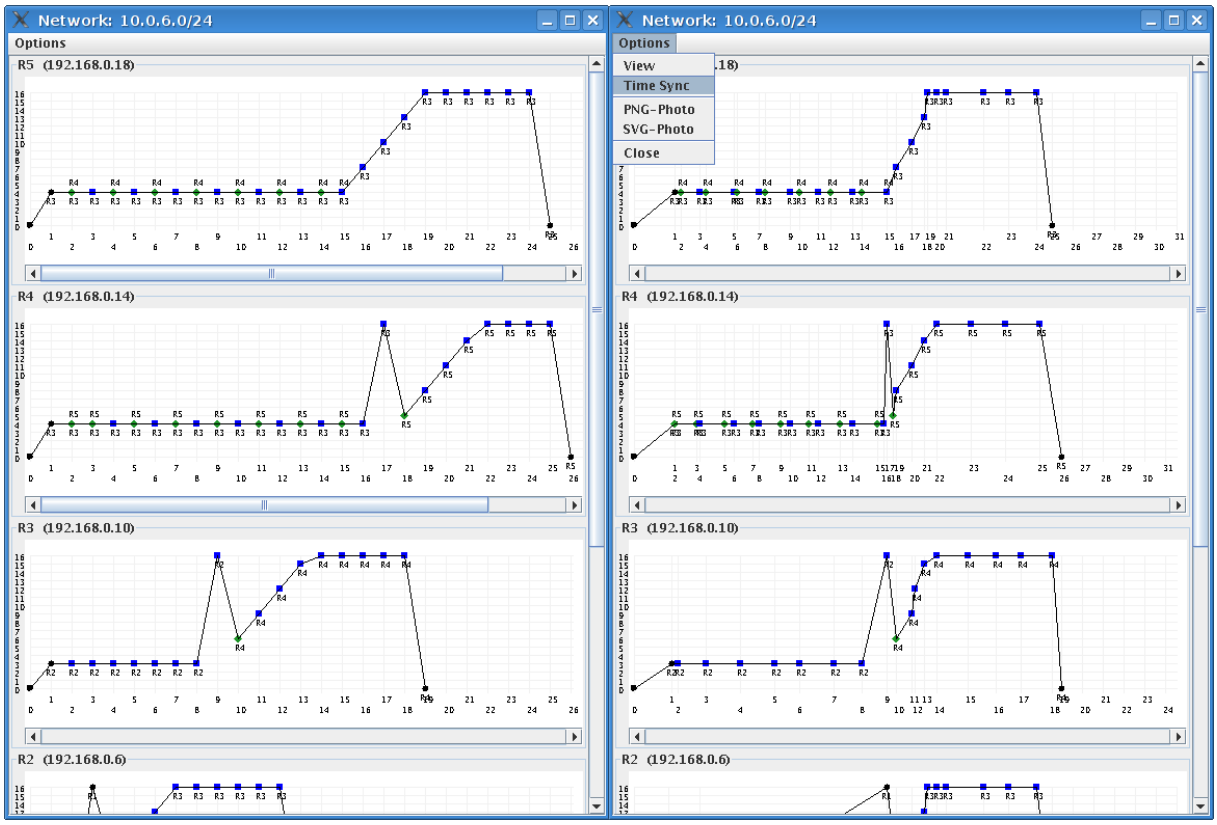


Abbildung 4.12: Metrikgraph eines CTI: Ohne (links) und mit (rechts) zeitlicher Synchronisation

Zusehen sind jeweils alle Metrikgraphen der beteiligten RIP-MTI-Daemons zu dem ausgewählten Subnetz (hier: 10.0.6.0), also der Verlauf der Metrik der Route zu diesem Subnetz in den einzelnen RIP-MTI-Daemons. Jeder Graphpunkt steht dabei für eine im XTPeer eingetroffene SL-Log-Meldung und die enthaltene Information. Ein „Klick mit der Maustaste“ auf einen der Graphpunkte öffnet die zugehörige Debug-Ausgabe (siehe Kapitel 4.3.2). In der Titelleiste des Fensters steht der IP-Address-Bereich des Subnetzes. Der Hostname des Routers, sowie dessen zugehörige IP-Adresse im VNUML-Management-Netzwerk, steht im Titel des Rahmens des jeweiligen Metrikgraphen. Die Linie des Funktionsgraphen zeigt die Metrik der Route zu dem Subnetz an, die der RIP-Daemon zu diesem Zeitpunkt inne hatte. Die Notation unter den Graphpunkten steht für den Hostnamen des benachbarten Routers, von dem der RIP-Daemon die Route übernommen hat. Über den Graphpunkten sind die Hostnamen der Nachbarrouter aufgeführt, die ein Routing-Update mit einer alternativen Route zum Subnetz gesendet haben. Die Route wurde aber aus unterschiedlichen Gründen nicht übernommen, meist weil sie eine höhere Metrik als die vorhandene Route besitzt. Fehlt die Notation über dem Graphpunkt, so handelt es sich um eine Bestätigung der vorhanden Route vom Nachbarrouter und wurde der Übersichtlichkeit wegen weggelassen, da sie mit der Notation unter dem Graphpunkt übereinstimmt.
Die Graphpunkte haben verschiedene Farben und Formen, die abhängig von der Information der zugehörigen SL-Log-Meldung gewählt wurden. Dies dient nur der Übersichtlichkeit, um zu erkennen unter welchem Graphpunkt welche Information zu erwarten ist.

Die Farben und Formen bedeuten das folgende:

■ : Blaue Quadrate: (RIP-MTI: Sonstiges)
Dieser Graphpunkt zeigt an, das der Bereich der Implementierung des RIP-MTI-Algorithmus zwar betreten wurde, aber es nicht zur Überprüfung eines Simple- oder Source-Loops gekommen ist. Ein solcher Graphpunkt wird gezeichnet, wenn der Nachbarrouter eine von ihm übernommene Route bestätigt oder wenn ein Routing-Update mit Metrik RIP-Infinity (16) eintrifft. Auch wenn ein Routing-Update mit einer alternative Route von einem anderen Nachbarrouter und einer kleineren Metrik als die Metrik, die die vorhandene Route hat, eintrifft, wird ein blaues Quadrat gezeichnet.

◆◆: Dunkel- und hellgrüne Rauten: (RIP-MTI: X-/Y-Test Simple-Loop Erkennung)
Der RIP-MTI-Algorithmus wurde aktiv und hat die alternative Route überprüft. Da aber eine gültige Route in der Routing-Tabelle vorhanden ist, musste der RIP-MTI-Algorithmus nicht entscheiden, ob die neue Route ein gültiger Ersatz für die alte Route in der Routing-Tabelle ist. Es wurde lediglich überprüft, ob die Routen-Kombination aus der alternativen Route und der in der Routing-Tabelle vorhandenen Route einen Simple-Loop bildet. Hat der RIP-MTI-Algorithmus den minimalen Simple-Loop aktualisiert oder bestätigt, so wird eine hellgrüne Raute gezeichnet. Wurde ein Simple-Loop gefunden, der größer ist als der bereits in der MSILM-Tabelle an dieser Stelle eingetragene, so wird eine dunkelgrüne Raute gezeichnet.

△▽ : Gelbe Dreiecke und rote kopfstehende Dreiecke: (RIP-MTI: X-/Y-Test verwirft Route)
Diese beiden Graphpunkte treten immer zusammen auf und zeigen an, dass der RIP-MTI-Algorithmus aktiv wurde und die alternative Route abgelehnt hat. Die vorhandene Route in der Routing-Tabelle ist ausgefallen und hat die Metrik RIP-Infinity (16), was über das rote kopfüber stehende Dreieck angezeigt wird. Die angebotene, alternative Route hat der RIP-MTI-Algorithmus anhand deren Metrik als Source-Loop erkannt. Das gelbe Dreieck zeigt die Metrik der abgelehnten, alternativen Route an, die in konventionellen RIP-Routern die ausgefallene Route in der Routing-Tabelle direkt ersetzt hätte.

■ ■ : Gelbe und rote Quadrate: (RIP-MTI: Verwirft Route anhand weiterer Kriterien)
Wie bei den gelben und roten Dreiecken lehnt auch hier der RIP-MTI-Algorithmus es ab, die ausgefallene Route in der Routing-Tabelle durch die alternativ angebotene Route zu ersetzen. Diese Graphpunkte zeigen an, das der RIP-MTI-Algorithmus eine Route ablehnt, allerdings nicht über den X-/Y-Test, sondern aus anderen Gründen. Diese Graphpunkte sind bislang eigentlich nur beim RIP-MTI-Careful-Mode und beim LMIDT anzutreffen. Hierbei wird die alternative Route aufgrund weiterer Kriterien solange abgelehnt, bis ihre Gültigkeit feststeht. Die gelben Quadrate zeigen die Metrik der abgelehnten Route an, während die roten Quadrate die derzeitige Metrik, RIP-Infinity, der in der Routing-Tabelle vorhandenen Route anzeigen.

● : Schwarze Punkte: (Keine Funktionen des RIP-MTI)
Wenn der Bereich der Implementierung des RIP-MTI-Algorithmus nicht betreten wird, werden schwarze Punkte gezeichnet. Dies ist gegenwärtig dann der Fall, wenn der RIP-MTI-Algorithmus des RIP-MTI-Daemons ausgeschaltet wurde oder aber auch, wenn vom gleichen Nachbarrouter, von dem die bisherige Route übernommen wurde, nun eine Route zum gleichen Subnetz mit einer kleinerer Metrik angeboten wird. In diesem Fall wird der RIP-MTI-Algorithmus nicht aufgerufen. Derzeit gibt es fünf mögliche Fälle für schwarze Punkte :

1) Der RIP-MTI-Algorithmus ist abgeschaltet. In diesem Fall werden im Metrikgraphen ausschließlich schwarze Punkte gezeichnet, da die notwendigen Informationen fehlen.
2) Der Timeout-Timer ist abgelaufen. Das Subnetz gilt nun als Unerreichbar und die Metrik der Route wird auf RIP-Infinity (16) gesetzt.
3) Der Garbage-Timeout-Timer ist abgelaufen. Die Route zum unerreichbaren Subnetz wird nun komplett aus der Routing-Tabelle gelöscht. Im XTPeer wird dieser Vorgang dadurch angezeigt, dass die Metrik der Route auf 0 gesetzt wird.
4) Der Router, von dem die Route zum Subnetz übernommen wurde, sendet nun eine Route zum gleichen Subnetz mit einer niedrigeren Metrik, die sofort übernommen wird.
5) Bei einem Fehler des Message-Parsers, der aus der MSG_MTI_DEBUG_INFO-Nachricht die Informationen auswertet. Da die SL-Protokoll-Syntax im RIP-MTI-Daemon und im XTPeer jeweils fest implementiert ist, muss der XTPeer zum RIP-MTI-Daemon passen.

Im Menüpunkt Options des Metrikgraph-Fensters sind folgende Einstellungen zu finden:

- View: Dieser Menüpunkt bietet die Möglichkeit einzustellen, ob und in welcher Reihenfolge die Metrikgraphen der RIP-Daemons angezeigt werden. View öffnet ein Fenster mit zwei Listen, wobei die obere Liste alle RIP-Daemons beinhaltet, während die untere Liste nur die RIP-Daemons, in der Reihenfolge, beinhaltet, die angezeigt werden. Mit Hilfe der Maus können die RIP-Daemons per „Drag&Drop“ aus der oberen Liste in die untere gezogen werden oder die Reihenfolge der RIP-Daemons in der unteren Liste verändert werden. Mit der rechten Maustaste kann der selektierte RIP-Daemon aus der unteren Liste gelöscht werden. Somit können die Metrikgraphen uninteressanter RIP-Daemons zu Gunsten der Übersichtlichkeit ausgeblendet werden.

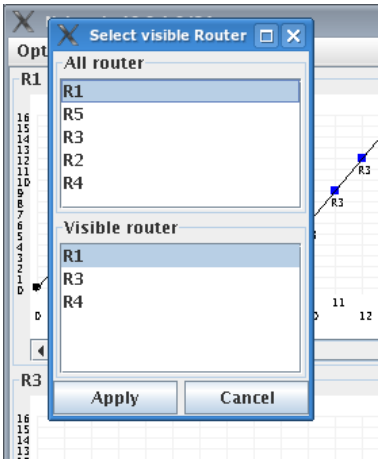


Abbildung 4.13: Der View-Dialog

- TimeSync: Mit dieser Einstellung können die Abstände der Graphpunkte nach den zeitlichen Abständen der zugehörigen SL-Log-Meldungen synchronisiert werden. RIP-Daemons die von mehreren verschiedenen Routern Routing-Updates mit alternativen Routen zum Subnetz erhalten, erzeugen mehr SL-Log-Meldungen als RIP-Daemons, die keine oder kaum Routing-Updates mit alternativen Routen erhalten. Daher ist es schwierig zwei Metrikgraphen einer Route zu einem Subnetz verschiedener RIP-Daemons direkt zu vergleichen. Die TimeSync Funktion synchronisiert die verschiedenen Metrikgraphen nach der Zeit der Generierung der zugehörigen SL-Log-Meldungen und erlaubt so zur Vergleichung eine einfachere aber auch unübersichtlichere Betrachtung des Verhaltens aller RIP-Daemons, bzgl. ihrer Routing-Updates.

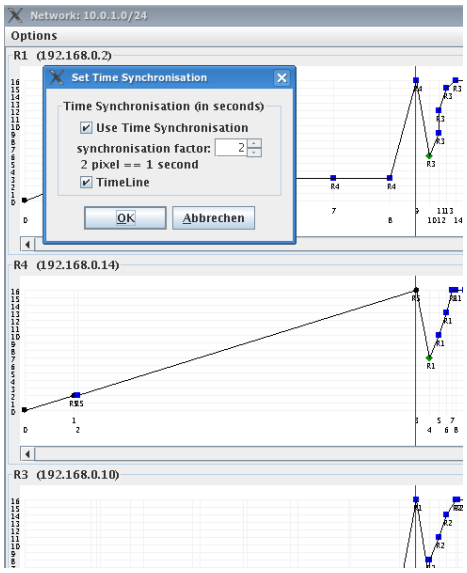


Abbildung 4.14: Der TimeSync-Dialog

- PNG-/SVG-Photo: Mit Hilfe dieser beiden Photo-Funktionen können Bildschirmphotos der angezeigten Metrikgraphen, in den jeweiligen Formaten, erstellt werden. Die in Kapitel 5 verwendeten Abbildungen der Metrikgraphen sind PNG-Bilder, die mit Hilfe dieser Funktion direkt aus dem Programm erstellt wurden.

4.3.4 Speichern und Laden protokollierter Daten

Der XTPeer bietet die Möglichkeit die während eines Testdurchlaufs protokollierten Datensätze zu speichern. Die Daten können im CSV[13]-Format zur Bearbeitung mit externen Programmen, wie zum Beispiel Tabellenkalkulations-Programmen, gespeichert werden. Darüber hinaus können die Daten auch in dem XTPeer eigenen, auf XML-basierenden, SDF[14]-Format gespeichert werden. Dieses Format bietet die Möglichkeit den Datensatz wieder in den XTPeer zu laden, um ihn mit den Datensätzen anderer Testdurchläufe zu vergleichen. Die Möglichkeit des Speicherns und Ladens der gesammelten Daten wurde bereits von S. Lange in [Lan07] implementiert und ist dort näher erläutert. Insbesondere das SDF-Format wurde für die Speicherung der zusätzlichen Informationen des RIP-MTI-Algorithmus erweitert. Zudem wurde hier der Metrikgraph und die Debug-Ausgabe in die grafische Anzeige der Daten eingebunden.

4.3.5 Der Interface-Status AUTOTRIGGER

Der Status eines jeden Interface, der in der Testumgebung eingebunden Rechner, kann über den XTPeer eingestellt werden. Der Status legt fest, ob der RIP-Daemon über dieses Interface Routing-Updates sendet, oder nicht. Die Funktion wurde bereits von D. Pähler in [Päh06] mit der Auswahl zwischen AUTO und MANUAL implementiert. In dieser Arbeit wurde der Status AUTOTRIGGER hinzugefügt. Ist das Interface im Status MANUAL, so sendet der RIP-Daemon keinerlei Routing-Updates an den Nachbarrouter jenseits des Interfaces. Nur manuell getriggerte Routing-Updates, die vom XTPeer aus, über einen „Klick“ auf das Interface,

13 comma separated files
14 scenario data files

ausgelöst wurden, sind zugelassen. Sämtliche vom RIP-Daemon selbständig generierten Routing-Updates werden unterdrückt und nicht über dieses Interface gesendet. Ist das Interface im Status AUTOTRIGGER, so wird der Update-Timer und damit die periodischen Routing-Updates unterdrückt. Routing-Updates die aufgrund einer Veränderung in der Routing-Tabelle erzeugt wurden, die sogenannten Triggered-Updates (siehe Kapitel 2.3.1), sind zugelassen und werden an den Nachbarrouter gesendet. Dieser Status ist vor allem für die aktiven RIP-MTI-Modes interessant, die selbständig Routing-(Triggered-)Updates aussenden, um zum Beispiel einen vorhandenen Source-Loop in einer angrenzenden Netzwerkschleife zu löschen. Stehen die Interfaces im Status AUTO, so wird der RIP-Daemon zwar auch in keinster Weise beim Senden von Routing-Updates behindert, aber ein vorhandener Source-Loop wird so eventuell bereits durch ein periodisches Routing-Update gelöscht.
Auch der Generator im XTPeer, von T. Keupen, wurde um den Interface-Status AUTOTRIGGER erweitert. Um hier nach dem vordefiniertem Ablauf die Interfaces in den Status AUTOTRIGGER zu schalten, muss in der Konfigurationsdatei des Generators in der letzten Zeile der Buchstabe „t“, anstelle des Buchstaben „a“ für den Status AUTO, angegebenen werden. Weiter Informationen zum Generator und dessen Konfiguration findet sich in [Keu07].

4.3.6 Neu-Initialisierung einer Simulation mit RIP-Daemons

Die gesammelten und im XTPeer gespeicherten Daten einer VNUML-Simulation mit RIP-Daemons können über den Menü-Punkt „Reinitialize for new Simulation“ im Menü „Scenario“ des XTPeers gelöscht werden. Unabhängig davon können nun auch die in den Strukturen des RIP-MTI-Algorithmus gesammelten Informationen, die Daten in der MSILM- und MRPM-Tabelle, in den RIP-Daemons der Simulation gelöscht werden, entweder für jeden Daemon einzeln oder für alle über den Menü-Punkt „Clean RT and reset MTI“ im Menü „Scenario“ des XTPeers. Dabei wird die Routing-Tabelle des RIP-Daemons geleert und der RIP-MTI-Algorithmus wird neu initialisiert, so dass die Ausgangssituation des Netzwerks, wie zu Beginn des gesamten Routing-Prozesses, wieder vorliegt um eine Simulation erneut durchzuspielen.

4.4 Zusammenfassung

Die in diesem Kapitel vorgestellte Software stellt die Testumgebung für die Überprüfung der Funktionalität des in Kapitel 3 vorgestellten RIP-MTI-Algorithmus dar. Mit Hilfe der Netzwerksimulationssoftware VNUML lassen sich Linux-Netzwerke mit beliebiger Topologie auf einem Host-System aufbauen und mit beliebiger Software bespielen. Der RIP-Daemon der Software-Routing-Suite Quagga stellt eine hervorragende Ausgangsbasis für die Implementierung des RIP-MTI-Algorithmus dar. Die Routing-Daemons von Quagga werden bereits als günstige Router-Alternativen in einigen produktiven Netzwerksystemen eingesetzt, so dass auch auf die Funktionalität des Quagga RIP-Daemons vertraut werden darf und allein die Weiterentwicklung des RIP-MTI-Algorithmus im Vordergrund stehen kann. Gleichzeitig kann so auch demonstriert werden, dass der RIP-MTI-Algorithmus die vorhandene RIP-Spezifikation lediglich erweitert und nicht neu definiert und somit auch bestehende Implementierungen ausgebaut werden können. Der XTPeer ermöglicht die Ansteuerung und Analyse der RIP-Daemons und nun auch des RIP-MTI-Algorithmus. Zusammen mit VNUML lassen sich so schnell, beliebige Netzwerk-Situationen herstellen, in denen die Funktion des RIP-MTI-Algorithmus über den RIP-MTI-Daemon nachvollziehbar überprüft werden kann.

5 Weiterentwicklung und Implementierung

In diesem Kapitel werden Teile der Implementierung des neuen Quagga RIP-MTI-Daemons näher erläutert. Hier steht nicht der Quellcode im Vordergrund, sondern das Verhalten der Implementierung, insbesondere der in den vorherigen Kapiteln vorgestellten RIP-MTI-Modes. Das Verhalten des RIP-MTI-Daemons und seiner Einstellungen wird mit Hilfe des XTPeers analysiert. Die Abbildungen der Ausgaben des XTPeer in diesem Kapitel sind Bildschirmphotos einer entsprechenden Simulationen des beschriebenen Szenarios. Die Metrikgraphen wurden über die, im XTPeer implementierte, PNG-Photo-Funktion erstellt.
Die Implementierung des RIP-MTI-Daemons basiert zwar auf der von T. Koch aus [Koc05], ist allerdings aufgrund der zahlreichen Veränderungen nicht mehr mit dieser zu Vergleichen. Zum einen wurde die Implementierung für die Konfiguration und Analyse durch den XTPeer erweitert und angepasst und zum Anderen wurde auch gleich versucht, die durch die Möglichkeit der Analyse erkannten Schwächen mit entsprechenden Erweiterungen, bzw. Veränderungen, zu beheben.

5.1 Implementierung der Funktionen des RIP-MTI-Algorithmus

Die Implementierung erlaubt die Konfiguration der Funktionen des RIP-MTI-Algorithmus, sowie das Sammeln der bearbeiteten Informationen, mit Hilfe des XTPeers. Die Möglichkeiten der Konfiguration werden im Kapitel 4.3.1 behandelt und die zur Analyse gesammelten Daten in den Kapiteln 4.3.2 und 4.3.3 erläutert. Hier soll nur kurz auf die entsprechende Implementierung im RIP-MTI-Daemon eingegangen werden.

```
struct rip_mti_state{
  int mode;
  int careful_mode;
  int careful_state;
  int careful_value;
  int orientation;
  int activity;
  int lmidt;
  int lmidt_value;
  int lmidt_table;
  int runtimer;
  int runtimer_value;
  int autoconfiguration;
  int trigger_interval;
  int msilm_calc;
  int oldmetric_delaytimer;
  int oldmetric_delayvalue;
};
```

Über das C-Konstrukt „rip_mti_state“ in der Header-Datei „rip_mti.h“ wird die Konfiguration des RIP-MTI-Algorithmus verwaltet. Es handelt sich hierbei um die Zusammenfassung mehrerer Integervariablen, die an den entsprechenden Stellen in der Implementierung des RIP-MTI-Algorithmus abgefragt werden und abhängig von ihrem Wert, das weitere Verhalten bestimmen. Die Werte, die die Integervariablen annehmen können sind über die Präprozessoranweisung „#define“, ebenfalls in der Datei „rip_mti.h“, definiert und auf der nächste Seite abgebildet. Auf die gleiche Weise ist auch die „Version“ des RIP-MTI-Daemons implementiert, die es ermöglichen soll, unterschiedliche Implementierungen des RIP-MTI-Daemons über einen XTPeer anzusteuern. Auch „RIP_MTI_Infintiy“, die maximale Metrik der Simple-Loops, ist hier implementiert, sowie mit „RIP-MTI-Alive“ die Lebensdauer der in den Tabellen gespeicherten Simple-Loops. Wird ein Simple-Loop über die Zeit von RIP-MTI-Alive nicht mehr bestätigt, so wird dessen Metrik auf RIP-MTI-Infinity gesetzt. Die Zeit von RIP-MTI-Alive entspricht der Summe der eingestellten Zeit des Timeout-Timers und der Zeit des Garbage-Collection-Timers. In der Datei „rip_mti.h“ sind auch die Protokollwörter, die für die SL-Anbindung mit dem XTPeer benötigt werden, definiert. Der SL-Nachrichtentyp „RIP_SLC_MSG_MTI_DEBUG_INFO“, der in der Header-Datei „sl_client.h“ implementiert ist, beinhaltet die gesammelten Daten des RIP-MTI-Algorithmus. Die Protokollwörter, die für die XT-Anbindung des XTPeers benötigt werden, sind in der Datei „rip_xt.h“ implementiert.

Die folgenden Werte können die Integervariablen im C-Konstrukts „rip_mti_state“ annehmen.

```
//RIP MTI MODES (mode)
#define RIP_MTI_OFF      0 //no RIP_MTI Mode used
#define RIP_MTI_LISTEN   1 //listen only, no route decision
#define RIP_MTI_NORMAL   2 //normal mode
#define RIP_MTI_STRICT   3 //strict mode
#define RIP_MTI_CAREFUL  4 //careful mode

//RIP-MTI-Careful Modes (careful_mode)
#define RIP_MTI_CAREFUL_ESHC 0    //External Split Horizon Check
#define RIP_MTI_CAREFUL_DT   1    //Deny Timer
#define RIP_MTI_CAREFUL_RT   2    //Request Timer

//MTI CAREFUL ESHC MODE (careful_state)
#define RIP_MTI_CAREFUL_ESHC_PASSIVE   0 //stay calm
#define RIP_MTI_CAREFUL_ESHC_ACTIVE    1 //send request

//RIP MTI ACTIVITY (activity)
#define RIP_MTI_PASSIVE           0 //stay calm
#define RIP_MTI_ACTIVE            1 //send response

//RIP MTI ORIENTATION (orientation)
#define RIP_MTI_INTERFACE       0 //align with interfaces
#define RIP_MTI_IP_MULTICAST    1 //align with IPs(send via interfaces)
#define RIP_MTI_IP_UNICAST      2//align with IPs(send to neighbour IP)
#define RIP_MTI_IP_IF_MULTICAST 3 //align with IPs and interfaces
#define RIP_MTI_IP_IF_UNICAST   4 //align with IPs and interfaces

//RIP MTI TIMER (careful_state, runtimer, oldmetric_denytimer, lmidt)
#define RIP_MTI_TIMER_OFF       0 //don't use a timer
#define RIP_MTI_TIMER_VALUE     1 //a fix timer value (in seconds)
#define RIP_MTI_TIMER_MAXMSILM  2//multiply timerval with largest msilm
#define RIP_MTI_TIMER_MRPM      3 //multiply timerval with mrpm
#define RIP_MTI_TIMER_UPDATE    4 //multiply timerval with update-timer

//RIP MTI (AUTO)-CONFIGURATION (autoconfiguration)
#define RIP_MTI_AUTOCONFIG_OFF  0 //mti's autoconfiguration is off
#define RIP_MTI_AUTOCONFIG_ON   1 //mti tries to configure itself and

//MSILM-Table computation (msilm_calc)
#define RIP_MTI_MSILM_RECV 0  //fill MSILM table only by received routes
#define RIP_MTI_MSILM_CALC 1  //calculate MSILM table entries

//LMIDT use MSILM or MRPM table (lmidt_table)
#define RIP_MTI_USE_MSILM    0 //lmidt uses msilm table
#define RIP_MTI_USE_MRPM     1 //lmidt uses mrpm table
```

5.2 Die MRPM-Tabelle

In der MRPM-Tabelle sind die MRPM-Werte (Minimal-Return-Path-Metric) der einzelnen Interfaces gespeichert. Diese Metriken beschreiben die Distanz des kürzesten Weges, den ein über das Interface ausgesendetes Daten-Paket zurücklegen müsste, um über ein anderes Interface wieder zum RIP-Daemon zurückzukehren. Die MRPM-Tabelle ist Bestandteil der Kernfunktionalität des RIP-MTI-Algorithmus und wird für den X-/Y-Test zur Erkennung

weiterer Simple-Loops benötigt. Die MRPM-Tabelle ist als Vektor implementiert. Quagga stellt eine eigene Vektor-Struktur, sowie die benötigten Funktionen für den Zugriff, zur Verfügung. Die Einträge in der MRPM-Tabelle sind vom Typ „mti_mrpm_entry“, einem in der Header-Datei „rip_mti.h“ definierten C-Konstrukt. Ein Eintrag in der MRPM-Tabelle beinhaltet die folgenden Informationen:

```
struct mti_mrpm_entry
{
 struct in_addr ip;
 unsigned int ifindex;
 time_t timer;
 struct thread *t_send;
 struct thread *t_request;
 unsigned int index;
 struct sockaddr_in *sockip;
 struct mti_msilm_entry *mrpm;
};
```

In dem „mti_mrpm_entry“-Konstrukt ist die Learned-From-IP-Adresse (LF-IP-Adresse) des benachbarten Routers, von dem die Routing-Updates ausgehen, sowie das zugehörige Interface in den Variablen „ip“ und „ifindex“ gespeichert. Die Variable „timer“ beinhaltet den Zeitpunkt des letzten Zugriffs auf einen Eintrag und wird benötigt, um ihn aus der MRPM-Tabelle zu löschen, falls sich die Erreichbarkeiten im Netzwerk verändert haben und die beinhaltete Information nicht mehr gültig ist. Es sind zwei Thread-Variablen implementiert, „t_send“ und „t_request“, die verzögertes (gebündeltes) senden von RIP-Response- bzw. Request-Nachrichten ermöglichen sollen. Erzeugt der RIP-MTI-Algorithmus eine RIP-Nachricht, um zum Beispiel die Konvergenz des Netzwerks zu beschleunigen, so wird die Nachricht nicht sofort gesendet, sondern der jeweilige Thread wird über eine bestimmte Zeitspanne gestartet und erst nach Ablauf des Threads wird eine einzige Nachricht gesendet, die alle in diesem Zeitraum anfallenden Veränderungen berücksichtigt. Dies entspricht vom Prinzip her dem fünf Sekunden Intervall bei Triggerd-Updates (siehe Kapitel 2.3.1) zur Reduzierung des RIP-Nachrichten-Verkehrsaufkommens. Um zwei Einträge der MRPM-Tabelle voneinander unterscheiden zu können werden die Positionsindizes der Einträge verwendet. Um auch unabhängig von der MRPM-Tabelle auf den Positionsindex zugreifen zu können wird er unter der Variable „index“ im „mti_mrpm_entry“-Konstrukt gespeichert. Bei der Variable „sockip“ handelt es sich um eine Hilfsvariable, die für das Senden von RIP-Nachrichten via Unicast benötigt wird. Beim Senden via Multicast wird diese Variable nicht verwendet, sondern das dazugehörige Interface. Der MRPM-Wert selbst wird nicht direkt im „mti_mrpm_entry“-Konstrukt gespeichert, sondern ist mit der Variable „mrpm“ über eine Referenz (Pointer) auf ein „mti_msilm_entry“-Konstrukt aus der MSILM-Tabelle verknüpft, da jeder MRPM-Wert eines Interfaces letztlich ein MSILM-Wert zwischen zwei Interfaces ist. Bei der Initialisierung eines Eintrags in der MRPM-Tabelle ist der MRPM-Wert selbst noch nicht bekannt. Darum wird auf das spezielle „mti_msilm_entry“-Konstrukt „mrpm_null_entry“ verwiesen, das den MRPM-Wert RIP-MTI-Infinity (31) besitzt. Erst nach der Erkennung eines Simple-Loops wird dann eine Referenz auf den entsprechenden Eintrag in der MSILM-Tabelle gesetzt, ansonsten verbleibt die Referenz auf „mrpm_null_entry“.
Vom Prinzip her funktioniert die MRPM-Tabelle wie eine Hash-Tabelle mit den Interfaces als identifizierende Schlüssel. Soll sich der RIP-MTI-Algorithmus über einem anderen Objekt statt der Interfaces orientieren, zum Beispiel über die LF-IP-Adresse, so muss lediglich der Schlüssel der MRPM-Tabelle ausgetauscht werden. Jeder Eintrag in der MRPM-Tabelle steht für eine Richtung, aus der alternative Routen angeboten werden könnten oder bereits eingetroffen sind.

Trifft ein Routing-Update mit einer alternativen Route zu einem Subnetz über ein anderes Interface beim Router ein, so wird das Interface der vorhandenen und der alternativen Route mit den eingetragenen Interfaces in der MRPM-Tabelle verglichen. Bei einer Übereinstimmung wird der Positionsindex des vorhanden Eintrags zurückgegeben und der Timer dieses Eintrags wird auf die Zeit des aktuellen Zugriffs gesetzt. Gibt es für ein Interface keine Übereinstimmung so wird ein neuer Eintrag für dieses Interface in der MRPM-Tabelle angelegt. Wurde ein Eintrag für eine längere Zeit nicht mehr abgerufen, weil zum Beispiel ein Interface abgeschaltet wurde oder weil sich die Erreichbarkeiten im Netzwerk verändert haben, so kann dies anhand der Differenz der letzten Zugriffszeit des Eintrags, aus der Variable „timer“, mit der gegenwärtigen Zeit, die über die Funktion „time(0)“ ermittelt wird, festgestellt werden. Der Eintrag kann gelöscht werden, wenn die Differenz einen bestimmten Wert, der aber bislang noch nicht festgelegt wurde, überschreitet, um ihn nicht ewig in der MRPM-Tabelle gespeichert zu halten. Die eigentliche Löschung ist in der XTPeer-Version aufgrund der Analysefunktion nicht implementiert. Alle Einträge der MRPM-Tabelle können über den Reset-Button („Clean RT and reset mti“) über den XTPeer gelöscht werden (siehe Kapitel 4.3.6).
Die MRPM-Tabelle wächst linear, hat ihre maximale Größe aber recht früh erreicht und wächst dann nicht unbedingt mit der Vergrößerung des Netzwerks, da die Anzahl der Einträge letztlich von der Anzahl der Interfaces des Routers, bzw. bei einer Orientierung über die LF-IP-Adresse, von den Routern und Verbindungen in direkter Nachbarschaft, abhängig ist. Der Speicheraufwand für die MRPM-Tabelle ist daher sehr gering. Im folgenden ist die Funktion „if2index“ aus der Datei „rip_mti.c“ aufgeführt, die Einträge für MRPM-Tabellen mit einer Orientierung über Interfaces zurückgibt, bzw. erzeugt. Bei einer Orientierung über die LF-IP-Adresse muss hier lediglich die IF-Abfrage nach dem Interface-Index „ifindex“ durch eine Abfrage nach der LF-IP-Adresse „from“ ausgetauscht werden, was mit der Funktion ip2index bereits implementiert ist.

```
int if2index(struct in_addr from, unsigned int ifindex){
 int index = -1;
 unsigned int i=1;
 struct mti_mrpm_entry *mrpm_entry;
 if(vector_active (mrpm_table) > 0){
    for (i = 0; i < vector_active(mrpm_table); i++){
       if ((mrpm_entry = vector_lookup(mrpm_table, i)) != NULL){
          if (mrpm_entry->ifindex==ifindex){
             index=i;
             mrpm_entry->timer=time(0);
          }
       }
    }
 }
 if (index == -1 ){
    struct mti_mrpm_entry *new_entry;
  new_entry=XMALLOC(MTYPE_RIP_MTI_ENTRY,sizeof(struct mti_mrpm_entry));
    IPV4_ADDR_COPY(&new_entry->ip,&from);
    new_entry->ifindex=ifindex;
    new_entry->sockip=createSockAddr_In(from);
    new_entry->timer=time(0);
    new_entry->mrpm=mrpm_null_entry;
    index = vector_set(mrpm_table,new_entry);
    new_entry->index=(index+1);
 }
 return (index+1);
}
```

5.3 Die MSILM-Tabelle

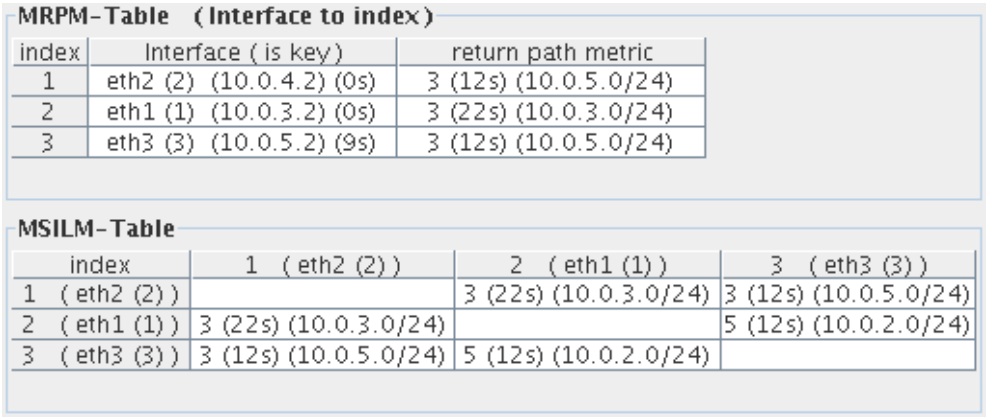
In der MSILM-Tabelle sind die MSILM-Werte (Minimal-Simple-Loop-Metric) gespeichert, also die Simple-Loops zwischen zwei Interfaces, bzw. Nachbarroutern bei einer Orientierung über die LF-IP-Adresse, mit der kleinsten Metrik. Die Tabelle orientiert sich an den Positionsindizes der Einträge aus der MRPM-Tabelle. Wie die MRPM-Tabelle ist auch die MSILM-Tabelle als Vektor implementiert. Die Einträge sind hier vom Typ „mti_msilm_entry“, einem C-Konstrukt, das in der Datei „rip_mti.h“ definiert ist und die folgenden Informationen enthält:

```
struct mti_msilm_entry
{
  u_int32_t metric;
  time_t timer;
  struct in_addr sil_prefix;
  int sil_mask_len;
  int msilm_calc;
};
```

In dem „mti_msilm_entry“-Konstrukt ist der eigentliche MSILM-Wert in der Variable „metric“ gespeichert. Die Variable „timer“ gibt den letzten Zeitpunkt einer Bestätigung des MSILM-Wertes an. Liegt der Zeitpunkt länger als RIP-MTI-Alive zurück, dann wird die Variable „metric“ auf auf RIP-MTI-Infinity (31) gesetzt. RIP-MTI-Alive setzt sich aus dem Timeout-Timer und dem Garbage-Collector-Timer zusammen. Vollständig gelöscht werden die Einträge derzeit nur über den Reset-Button im XTPeer (siehe 4.3.6). Die Variablen „sil_prefix“ und „sil_mask_len“ beinhalten das Subnetz, zu dem, über verschiedene Richtungen, Routing-Updates für zwei unterschiedliche Routen eingetroffen sind, wodurch der Eintrag in der MSILM-Tabelle erzeugt wurde. Diese beiden Variablen dienen lediglich der besseren Analyse durch den XTPeer. Die Variable „msilm_calc“ bezieht sich auf die Vollständigkeit der MSILM-Tabelle (siehe Kapitel 3.4.3), die nicht immer gegeben ist. Diese Variable zeigt an, ob ein Eintrag berechnet wurde oder aufgrund existierender Routen erstellt worden ist.
Da in der MSILM-Tabelle, wie auch in der MRPM-Tabelle, nur die unmittelbare topologische Umgebung abgebildet ist, hat sie ihre maximale Größe sehr schnell erreicht und wächst dann nicht unbedingt weiter, wenn das Netzwerk weiter wächst.
Bereits A. Schmidt hat in [Sch99] vorgeschlagen die MSILM-Tabelle linear in einem Vektor zu speichern. Hierbei wird ihre symmetrische Eigenschaft genutzt. Die MSILM-Tabelle wächst deswegen nur linear, mit jedem erkannten Simple-Loop zwischen zwei Interfaces. Sind die vorhandenen Simple-Loops nur auf Interface-Paare verteilt, dann ist die MSILM-Tabelle sogar kleiner als die MRPM-Tabelle, da zwei Einträge aus der MRPM-Tabelle auf einen Eintrag in der MSILM-Tabelle referenzieren.
Mit der folgenden Formel wird, anhand der Positionsindizes der zugehörigen Einträge aus der MRPM-Tabelle, die Position in der linearen MSILM-Tabelle (Vektor) berechnet, an der der MSILM-Wert eingetragen wird.

$$\begin{aligned} &if\ (IndexA > IndexB)\quad position = \frac{indexA*(indexA-1)}{2} + indexB \\ &if\ (IndexA < IndexB)\quad position = \frac{indexB*(indexB-1)}{2} + indexA \\ &if\ (IndexA = IndexB)\quad continue \end{aligned} \tag{1}$$

Das folgende Beispiel in Abbildung 5.1 zeigt die MRPM- und MSILM-Tabelle des Routers R1 aus der Nested-Loop-Topologie in Abbildung 5.6.



MRPM-Table (Interface to index)

index	Interface (is key)	return path metric
1	eth2 (2) (10.0.4.2) (0s)	3 (12s) (10.0.5.0/24)
2	eth1 (1) (10.0.3.2) (0s)	3 (22s) (10.0.3.0/24)
3	eth3 (3) (10.0.5.2) (9s)	3 (12s) (10.0.5.0/24)

MSILM-Table

index	1 (eth2 (2))	2 (eth1 (1))	3 (eth3 (3))
1 (eth2 (2))		3 (22s) (10.0.3.0/24)	3 (12s) (10.0.5.0/24)
2 (eth1 (1))	3 (22s) (10.0.3.0/24)		5 (12s) (10.0.2.0/24)
3 (eth3 (3))	3 (12s) (10.0.5.0/24)	5 (12s) (10.0.2.0/24)	

Abbildung 5.1: Beispiel für die MSILM-Tabellen-Anzeige des XTPeer

Mit Hilfe der Formel aus (1) wird die MSILM-Tabelle wie folgt in einem Vektor gespeichert:

Position im Vektor	MSILM-Eintrag (mti_msilm_entry)
2	3 (22s) (10.0.3.0/24)
4	3 (12s) (10.0.5.0/24)
5	5 (12s) (10.0.2.0/24)

Die nicht aufgeführten Vektor-Positionen dazwischen, hier 0, 1 und 3, bleiben leer und nehmen daher auch keinen Speicherplatz weg.
Für eine linearisierte MSILM-Tabelle ergibt sich somit immer die folgende Verteilung der Positionen im Vektor (mit der Formel aus (1)).

Index	1 (eth2 (2))	2 (eth1 (1))	3 (eth3 (3))	4	5	...
1 (eth2 (2))	1	2	4	7	11	16
2 (eth1 (1))	2	3	5	8	12	17
3 (eth3 (3))	4	5	6	9	13	18
4	7	8	9	10	14	19
5	11	12	13	14	15	20
...	16	17	18	19	20	21

Über den Positionsindex eines Eintrags aus der linearen MSILM-Tabelle können die beiden MRPM-Indizes mit der folgenden While-Schleife auch wieder hergeleitet werden. Eine solche Berechnung der MRPM-Indizes befindet sich derzeit allerdings nur im XTPeer, zur dortigen tabellarischen Anzeige der MSILM-Tabelle, wie in Abbildung 5.1 dargestellt.

```
int count=1; int h=1;
while( vector_position >= count + h ){
   count += h++;
}
int indexA = vector_position + 1 - count;
int indexB = h;
```

Es ergeben sich einige Vorteile aus dieser Art der Speicherung, vor allem bezogen auf das einfache, lineare Auslesen und den geringen Speicheraufwand.
Für einen Durchlauf durch die MSILM-Tabelle anhand der Einträge der MRPM-Tabelle wächst der Aufwand allerdings quadratisch zu der Anzahl der Einträge und zusätzlich auch linear zur Anzahl der vom RIP-MTI-Algorithmus zu überprüfenden alternativen Routen.
Im folgenden ist die Implementierung des Durchlaufs durch die MSILM-Tabelle über die MRPM-Indizes abgebildet. Dieser Durchlauf ist in der Funktion „mti_sil_timeout()“ in der Datei „rip_mti.c“ implementiert und dient der Erkennung ungültiger Einträge in den Tabellen.

```
unsigned int i, j, pos;
time_t now = time(0);
struct mti_msilm_entry *msilm_entry, *msilmA, *msilmB;
struct mti_mrpm_entry *mrpm_entry, *mrpmA, *mrpmB;

for (i = 0; i < vector_active(mrpm_table); i++) {
   if ((mrpmA = vector_lookup(mrpm_table, i)) != NULL){
      msilmA = mrpmA->mrpm;

      for (j = 0; j < i; j++) {
         if ((mrpmB = vector_lookup(mrpm_table, j)) != NULL) {
            msilmB = mrpmB->mrpm;

            /* calculate msilm position */
            pos = (( i * ( i - 1 )) / 2) + j;

            if ((msilm_entry = vector_lookup(msilm_table, pos)) != NULL)
            {
               ...
            }
            ...
         }
      }
   }
}
```

Da die MSILM-Tabelle für den RIP-MTI-Strict-Mode und einige der darauf aufbauenden RIP-MTI-Careful-Modes nicht benötigt wird, könnte auf sie verzichtet werden. Die MSILM-Tabelle wird aber benötigt, wenn auch auf den RIP-MTI-Normal-Mode zurückgegriffen werden soll.

5.4 Der X- und Y-Test (X-/Y-Test)

Im folgenden ist die C-Funktion, die den X-/Y-Test beinhaltet, aufgeführt. Sie befindet sich in der Datei „rip_mti.c“. Der X-/Y-Test ist Bestandteil der Kernfunktionalität des RIP-MTI-Algorithmus. Er wird bei allen RIP-MTI-Modes zur Erkennung von Simple-Loops anhand der alternativen Routen verwendet, wenn die zugehörige Route aus der Routing-Tabelle nicht ausgefallen ist. Erkannte Simple-Loops werden in die MSILM-Tabelle eingetragen, sofern ihre Metrik minimal ist und zu den entsprechenden Einträgen in der MRPM-Tabelle verlinkt.
Beim RIP-MTI-Strict-Mode und den RIP-MTI-Careful-Modes wird der X-/Y-Test auch verwendet, um Source-Loops zu erkennen und das CTI-Problem zu verhindern. Dabei werden Source-Loops verhindert indem sie nicht als Simple-Loops erkannt werden, was jedoch in bestimmten Situationen ein zu strenges Kriterium ist und die Konvergenz des Netzwerks gegenüber konventionellen RIP-Routern stark verlangsamt. In diesem Fall erkennt der RIP-

MTI-Algorithmus den Simple-Loop nicht und lehnt die gültige, alternative Route ab. Das Hauptproblem dabei ist der X-Test, der sich an den kleinen MRPM-Werten der Interfaces orientiert. Wie in Kapitel 3.4.2 beschrieben kann der X-Test durch den Y-Test ersetzt werden, wenn Routen mit kleineren Metriken nach ihrem Ausfall länger im RIP-MTI-Algorithmus verbleiben und erst nach einer Verzögerungszeit durch Routen mit größeren Metriken ersetzt werden, so dass der RIP-MTI-Algorithmus länger auf die kleinere Metrik zugreifen kann. Diese verzögerte Übernahme gilt nicht für die Route in der Routing-Tabelle, somit kommt es hier zu keiner Verlangsamung der Konvergenz. Der X-Test ist aktiviert wenn die Variable „oldmetric_delay“ auf „RIP_MTI_TIMER_OFF“ steht, ansonsten ist der X-Test deaktiviert und nur der Y-Test überprüft die alternativen Routen.

```
int xy_combination_test(int isold, unsigned int indexA, unsigned int
        indexB, u_int32_t metricA, u_int32_t metricB, u_int32_t combi)
{
 struct mti_msilm_entry *rmeA, *rmeB;
 int x = 0; int z = 1; int y = 0; int eq = 0;
 u_int32_t ymrpm = 2;

 rmeA = rip_mti_mrpm_lookup(indexA);
 rmeB = rip_mti_mrpm_lookup(indexB);

 if (metricA == metricB)   eq = 1;

 else if (metricA > metricB)
 {
   if (rmeA->metric != RIP_MTI_INFINITY) ymrpm = rmeA->metric;

   /** no y combination found if condition is true */
   if (ymrpm > (metricA - metricB) ) y=1;
 }
 else if (metricA < metricB)
 {
   if (rmeB->metric != RIP_MTI_INFINITY) ymrpm = rmeB->metric;

   /** no y combination found if condition is true */
   if (ymrpm > (metricB – metricA)) y=1;
 }
 if((rmeA->metric<RIP_MTI_INFINITY)&&(rmeB->metric<RIP_MTI_INFINITY))
 {
   z=0;
   /** no x combination found if condition is true */
   if (rmeA->metric + rmeB->metric > combi) x=1;
 }
 if(mti_state->oldmetric_delaytimer==RIP_MTI_TIMER_OFF)
 {
   if( (y || eq) && (x || z)  ) { return 1;  }
 }
 else
 {
   if( y || eq ) {  return 1; }
 }
 /** There is an X- or Y-combination, the route should be rejected */
 return 0;
```

5.5 Die RIP-MTI-Modes

Die RIP-MTI-Modes beschreiben im wesentlichen nur verschiedene Implementierungen des RIP-MTI-Algorithmus. Sie unterscheiden sich vor allem in der Vorgehensweise bei der Erkennung und Verhinderung von Source-Loops und der Vermeidung des CTI-Problems. Die RIP-MTI-Modes werden aufgerufen, wenn ein Routing-Update für eine alternative Route über ein anderes Interface eintrifft, bzw. bei einer Orientierung über die LF-IP-Adresse von einem anderen Router, und die vorhandene Route zum Subnetz in der Routing-Tabelle des Routers ausgefallen ist, also deren Metrik RIP-Infinity (16) beträgt. Die RIP-MTI-Modes sollen überprüfen, ob die ausgefallene Route in der Routing-Tabelle mit der neu angebotenen, alternativen Route ersetzt werden darf oder ob ein Source-Loop vorliegt und bei Annahme der alternativen Route die Gefahr des CTI-Problems besteht. Wurde eine alternative Route vom RIP-MTI-Algorithmus für gültig befunden, so kann es trotzdem sein, dass aufgrund eines anderen Kriteriums des Routing-Prozesses die Route abgelehnt wird und sie somit letztlich nicht in die Routing-Tabelle übernommen wird.

In Abbildung 5.2 ist die gleiche Y-Topologie zweimal abgebildet. In der oberen Y-Topologie verfügen alle Router des Netzwerks über eine gültige Route zum Subnetz 10.0.1.0/24. In der unteren Y-Topologie ist die Verbindung zum Subnetz bei Router R5 ausgefallen und das CTI-Problem ist aufgetreten. Die Route führt in eine Routingschleife.

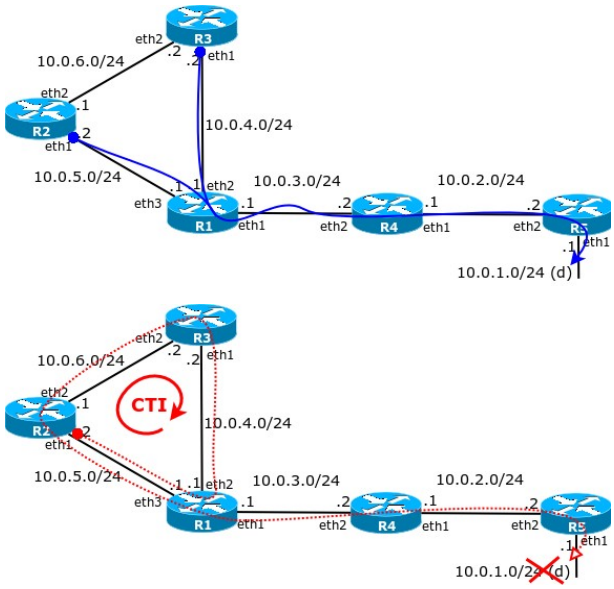


Abbildung 5.2: Route zum Subnetz in der Y-Topologie

In den beiden Metrikgraphen in Abbildung 5.3 sind die beiden möglichen Abläufe nach dem Ausfall der Route zum Subnetz 10.0.1.0/24 in der Y-Topologie aus Abbildung 5.2 zu sehen.

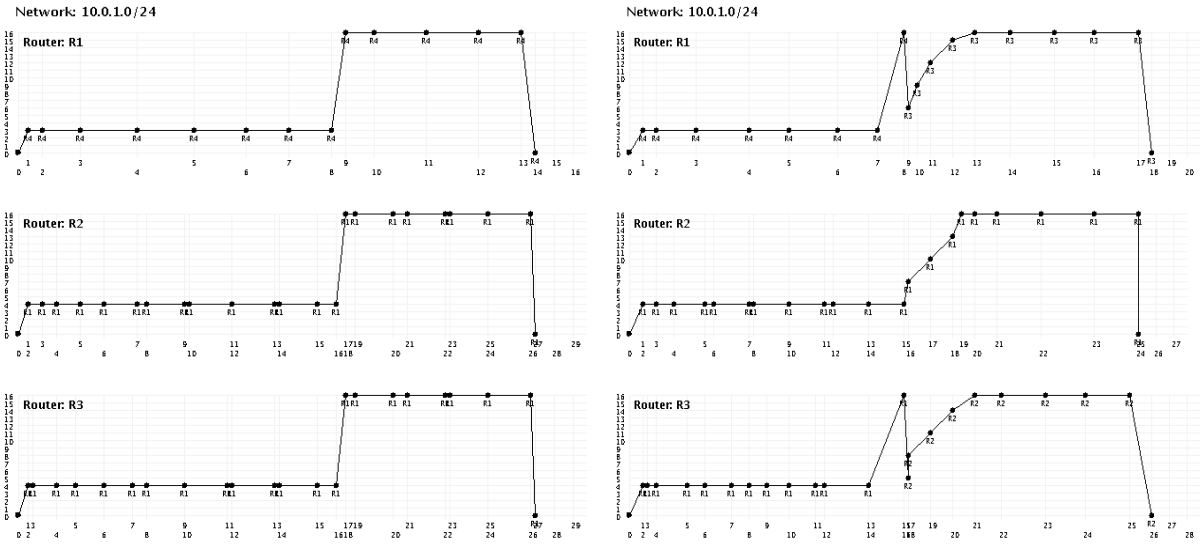


Abbildung 5.3: Wünschenswerter Ausfall-Ablauf (links) und Ausfall mit CTI in Folge (rechts)

Im linken Metrikgraph ist der erwünschte Ablauf nach Ausfall der Route zum Subnetz abgebildet. Die Information der Unerreichbarkeit des Subnetzes verteilt sich gleichmäßig im ganzen Netzwerk und alle Router in der Netzwerkschleife der Y-Topologie verlieren die Route unmittelbar zur gleichen Zeit.
Im Rechten Metrikgraphen ist das CTI-Problem zu sehen. Hier erreicht Router R2 die Information über die Unerreichbarkeit des Subnetzes 10.0.1.0/24, die er von Router R1 übernommen hatte, nicht. Router R2 sendet seine bisherige Route zum Subnetz mit der mittlerweile veralteten Information und falscher Metrik an Router R3, der sie als alternative Route annimmt und seine ausgefallen Route in der Routing-Tabelle ersetzt. Router R3 sendet die Route weiter an R1. Router R1 nimmt die Route von R3 an und sendet sie ebenfalls weiter. So kommt es zum CTI-Problem und es bildet sich eine Routingschleife über der Route.
Wäre die Metrik der Routen nicht auf den relativ kleinen Wert 16 für RIP-Infinity begrenzt, so würde das CTI-Problem die Konvergenz des Netzwerks noch stärker hinauszögern und weitaus größere Probleme verursachen.

5.5.1 RIP-MTI-Normal-Mode

Der RIP-MTI-Normal-Mode vergleicht die Routen-Kombination aus der alten, vorhandenen Route und der neuen, alternativen Route mit der MSILM-Metrik der beiden, den Routen zugehörigen, Interfaces A und B aus der MSILM-Tabelle. Da statt der Interfaces auch die LF-IP-Adresse verwendet werden kann sind diese hier als „indexA“ und „indexB“ aufgeführt, was dem Positionsindex des zugehörigen Eintrags aus der MRPM-Tabelle entspricht.

```
u_int32_t combi = metricA + metricB - 1;
struct mti_msilm_entry *msilm_entry=NULL;
msilm_entry = rip_mti_msilm_table_lookup(indexA,indexB);

return (combi>=msilm_entry->metric);
```

Der RIP-MTI-Normal-Mode gibt „true“ zurück, wenn ein Simple-Loop zwischen den beiden, zugehörigen Interfaces vorhanden ist, dessen Metrik kleiner oder gleich der Metrik der Routen-Kombination ist. Sind die Metriken gleich, dann beschreibt die Routen-Kombination den gespeicherten minimalen Simple-Loop. Ist kein Simple-Loop zwischen den beiden Interfaces vorhanden, so steht in der Variable „msilm_entry->metric“ der Initialisierungswert RIP-MTI-Infinity (31). Es gibt keine gültige Routen-Kombination deren Metrik größer RIP-MTI-Infinity ist. Der RIP-MTI-Normal-Mode gibt somit „false“ zurück und die alternative Route wird abgelehnt, wenn kein Simple-Loop vorhanden ist.

In Abbildung 5.4 lehnt der RIP-MTI-Normal-Mode in Router R1 die alternative Route zum Subnetz 10.0.1.0/24 über Interface eth2 ab, da zwischen seinen beiden Interfaces eth2 und eth1 kein Simple-Loop existiert. Somit wird hier der Source-Loop erkannt und das CTI-Problem verhindert.

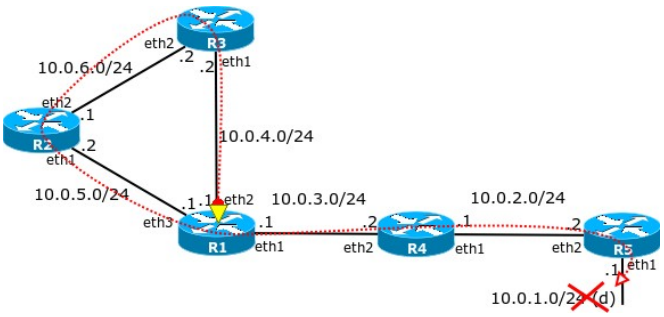


Abbildung 5.4: Der RIP-MTI-Normal-Mode in Y-Topologie

Es gilt : $m_{eth2}^{R1,d} + m_{eth1}^{R1,d} - 1 \geq msilm_{eth2,eth1}^{R1}$
$\rightarrow \; 6+3-1 \geq 31\,(f)$

Der folgende Metrikgraph in Abbildung 5.5 zeigt den Verlauf der Metrik der Route zum Subnetz 10.0.1.0/24 des Routers R1. Die Abbildung rechts neben dem Metrikgraphen ist ein Bildschirmphoto eines Ausschnitts der zugehörigen Debug-Ausgabe des entsprechenden Routing-Updates (siehe Abbildung 4.11 in Kapitel 4.3.2). Der RIP-MTI-Normal-Mode lehnt die an dieser Stelle von R3 angebotene, alternative Route mit der Metrik 6 ab, da der MSILM-Wert 31 nicht kleiner oder gleich der Routen-Kombination mit Metrik 8 ist.

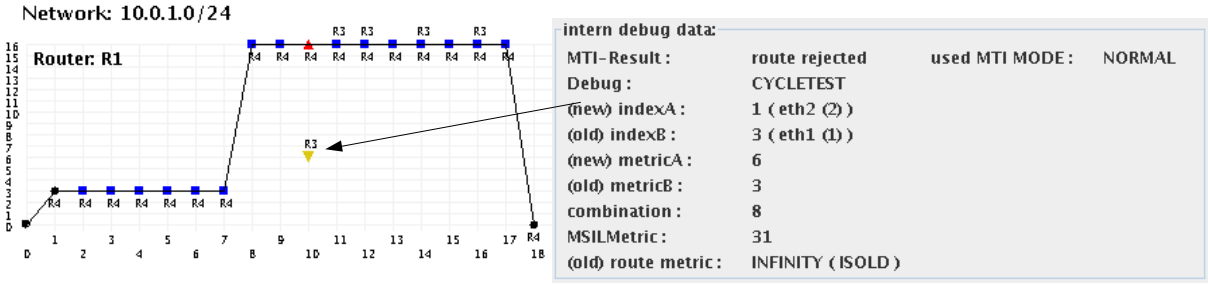


Abbildung 5.5: XTPeer Ausgabe des RIP-MTI-Normal-Mode

In komplexeren und stark vermaschten Topologien mit „verschachtelten Schleifen“, in denen ein Interface auch mit mehr als einem weiteren Interface einen Simple-Loop besitzt, genügt der RIP-MTI-Normal-Mode nicht mehr um das CTI-Problem zu verhindern.

In Abbildung 5.6 wurde die Topologie um eine Verbindung zwischen R3 und R4 erweitert, so dass Router R1 auch zwischen seinen Interfaces eth1 und eth2 einen Simple-Loop besitzt. Router R1 kann hier mit Hilfe des RIP-MTI-Normal-Mode den Source-Loop nicht erkennen und somit auch das CTI-Problem nicht verhindern. Der RIP-MTI-Normal-Mode in R1 erkennt eine gültige, alternative Route aufgrund des vorhandenen Simple-Loops zwischen den Interfaces eth1 und eth2.

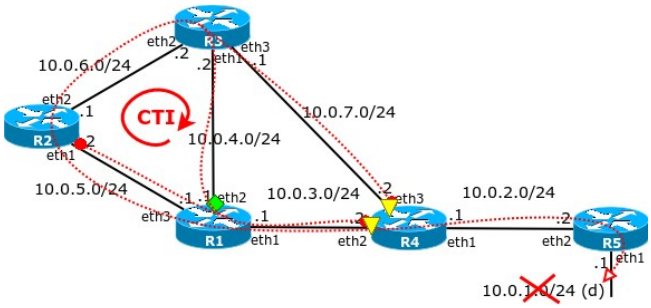


Abbildung 5.6: RIP-MTI-Normal-Mode bei Nested-Loops

Für Router R1 gilt: $m_{eth2}^{R1,d} + m_{eth1}^{R1,d} - 1 \geq msilm_{eth2,eth1}^{R1}$

$\rightarrow \; 6 + 3 - 1 \geq 3$

Router R4 kann zwar den Source-Loop erkennen und verhindern, dass das CTI-Problem, das nun von R3 und von R1 aus droht, auch über ihn auftritt, aber ansonsten kann R4 lediglich den Teil des Netzwerks hinter ihm vor den Auswirkungen des auftretenden CTI-Problems schützen.

Wäre hier nur die Verbindung zwischen Router R1 und R4 ausgefallen und das Subnetz 10.0.1.0/24 ansonsten erreichbar, dann könnte es sich bei der Route über R3 tatsächlich um eine gültige, alternative Route zum Subnetz handeln und dann würde der RIP-MTI-Normal-Mode in R1 richtig handeln.

Die beiden folgenden Metrikgraphen in Abbildung 5.7 zeigen den Verlauf der Metrik der Route zum Subnetz 10.0.1.0/24 in den Routern R1 und R4. In Router R1 findet das CTI-Problem statt und die Metrik wird bis RIP-Infinity hochgezählt. Router R4 kann den Source-Loop, der nun von Router R1 und R3 ausgeht, erkennen und verhindern.

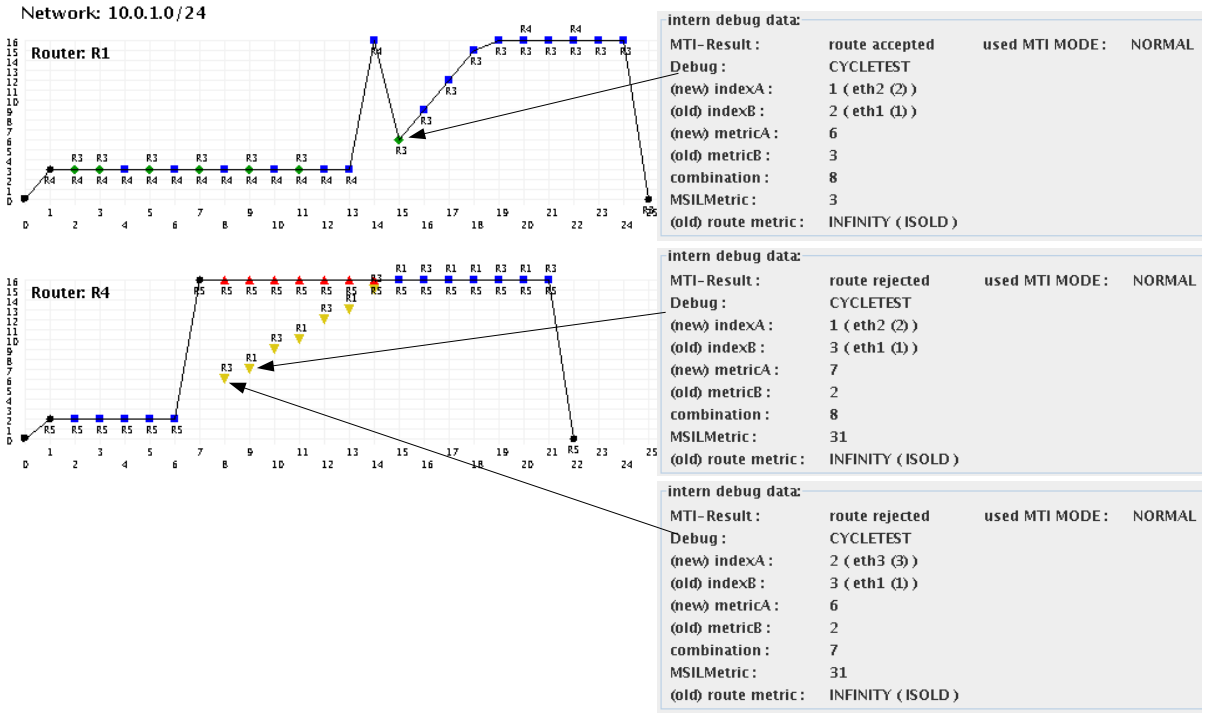


Abbildung 5.7: XTPeer Ausgabe des RIP-MTI-Normal-Mode bei Nested-Loops

Um auch in solchen Topologien das CTI-Problem vollständig zu verhindern, muss der in Kapitel 3.4.1 vorgestellte RIP-MTI-Strict-Mode verwendet werden.

5.5.2 RIP-MTI-Strict-Mode

Der RIP-MTI-Strict-Mode ist eine Erweiterung des RIP-MTI-Algorithmus, die auf dem Vorschlag von T. Kleemann aus [Kle01] basiert. Er unterscheidet sich gegenüber dem RIP-MTI-Normal-Mode nur während bei der Verhinderung von Source-Loops. Nach Ausfall einer Route überprüft der RIP-MTI-Strict-Mode jede alternative Route, die über ein anderes Interface angeboten wird, ob die Routen-Kombination mit der ausgefallenen Route aus der Routing-Tabelle den X-/Y-Test besteht. Ist das der Fall dann liegt das Subnetz auf einem Simple-Loop aus der Kombination dieser beiden Routen. Ist das nicht der Fall dann geht der RIP-MTI-Strict-Mode von einem Source-Loop aus und lehnt die alternative Route ab, weil deren Annahme zum CTI-Problem führen könnte. Der RIP-MTI-Strict-Mode verhinderte das CTI-Problem in allen bisherigen Test-Szenarien, allerdings werden auch viele gültige, alternative Routen unnötigerweise abgelehnt, was nach dem Ausfall einer Route die Konvergenz des Netzwerks trotz existierender, alternativer Router verlangsamt.

In Abbildung 5.8 erkennt der RIP-MTI-Strict-Mode in den Routern R1 und R4 jeweils den Source-Loop, der von Router R3 ausgeht. Das CTI-Problem wird hier verhindert. In diesem Beispiel ist der RIP-MTI-Strict-Mode allerdings nur mit dem Y-Test aktiv, der X-Test ist deaktiviert (unused), wie in der Debug-Ausgabe in Abbildung 5.9 zu sehen ist.

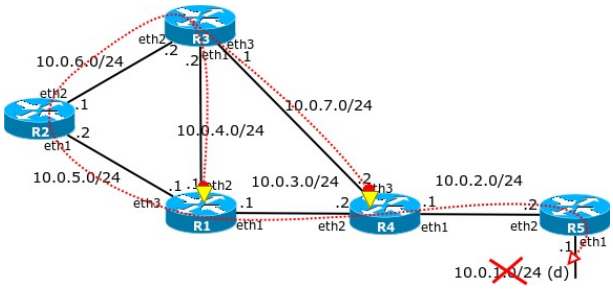


Abbildung 5.8: RIP-MTI-Strict-Mode bei Nested-Loops

Im folgenden ist jeweils der Metrikgraph der Router R1 und R4 für den Verlauf der Metrik der Route zum Subnetz 10.0.1.0/24 abgebildet.

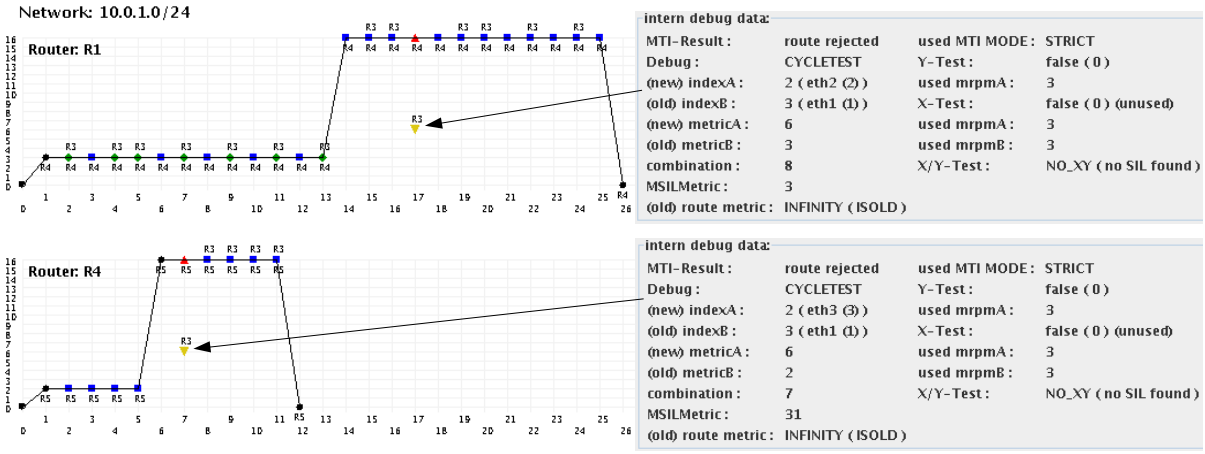


Abbildung 5.9: XTPeer Ausgabe des RIP-MTI-Strict-Mode bei Nested-Loops

Wäre im obigen Beispiel der X-Test aktiv, so würde bereits Router R3 den Source-Loop, ausgehend von R2, erkennen und die Route ablehnen. Allerdings würde R3 auch eine gültige, alternative Route über R1 ablehnen, falls die direkte Verbindung zwischen ihm und R4 ausgefallen wäre.

In Abbildung 5.10 ist der RIP-MTI-Strict-Mode mit dem X-Test aktiv. Router R3 kann die gültige, alternative Route zum Subnetz 10.0.1.0/24 weder über Router R1 noch über R2 annehmen, da der X-Test in beiden Fällen eine X-Kombination erkennt und die Annahme der Route verhindert.

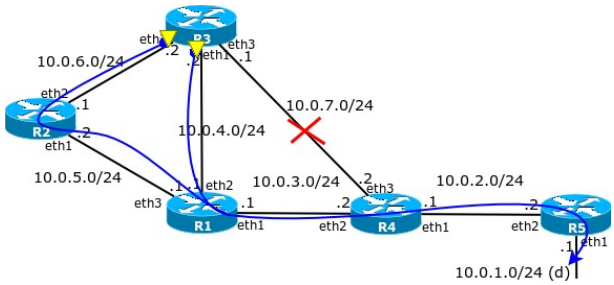


Abbildung 5.10: RIP-MTI-Strict mit X-Test

Für Router R3 berechnet der X-Test für die gültige, alternative Route von R1 und von R2 jeweils folgendes :

v. R1: $mrpm_{eth1}^{R3} + mrpm_{eth3}^{R3} > m_{eth3}^{R3,d} + m_{eth1}^{R3,d} - 1$
$\rightarrow \; 3+3>3+4-1(f)$

v. R2: $mrpm_{eth2}^{R3} + mrpm_{eth3}^{R3} > m_{eth3}^{R3,d} + m_{eth2}^{R3,d} - 1$
$\rightarrow \; 3+3>3+5-1(f)$

Das Beispiel verdeutlicht auch, dass weitere topologische Möglichkeiten für alternative Route nichts nützen und ebenfalls vom RIP-MTI-Strict-Mode abgelehnt werden.

Router R3 kann hier die gültige, alternative Route von Router R1 oder R2 erst annehmen, nachdem die ausgefallene Route zum Subnetz aus der Routing-Tabelle gelöscht worden ist, was allerdings erst mit Ablauf des Garbage-Collection-Timers, also nach 120 Sekunden, passiert. Wird der Ausfall der Route erst durch einen Timout festgestellt und die Metrik erst nachdem 180 Sekunden lang kein Routing-Update eingetroffen ist auf RIP-Infinity gesetzt, dann kann es aufgrund der Ablehnung der Route durch den RIP-MTI-Strict-Mode insgesamt bis zu 300 Sekunden (5 Minuten) dauern, bis der Router über die alternative Route zum Subnetz verfügt, die allerdings die ganze Zeit über bereits verfügbar war.

In der folgenden Abbildung ist der Metrikgraph für das Szenario aus Abbildung 5.10 dargestellt. In diesem Fall verliert Router R3 die Route zum Subnetz 10.0.1.0/24 durch einen Timeout, da kein Routing-Update mehr über die ausgefallene Verbindung von Router R4 eintrifft. Die

alternative Route über R1 wird 335 Sekunden (5,35 Minuten) nach dem letzten Routing-Update von R4 angenommen. Davon sind allerdings 180 Sekunden dem Timeout-Timer der Standard-RIP-Spezifikation geschuldet. Das Beispiel zeigt somit auch, dass die Timer des RIP-Algorithmus generell heruntergeregelt werden sollten, um den Ausfall einer Route schneller zu bemerken und früher auf alternative Routen zugreifen zu können.

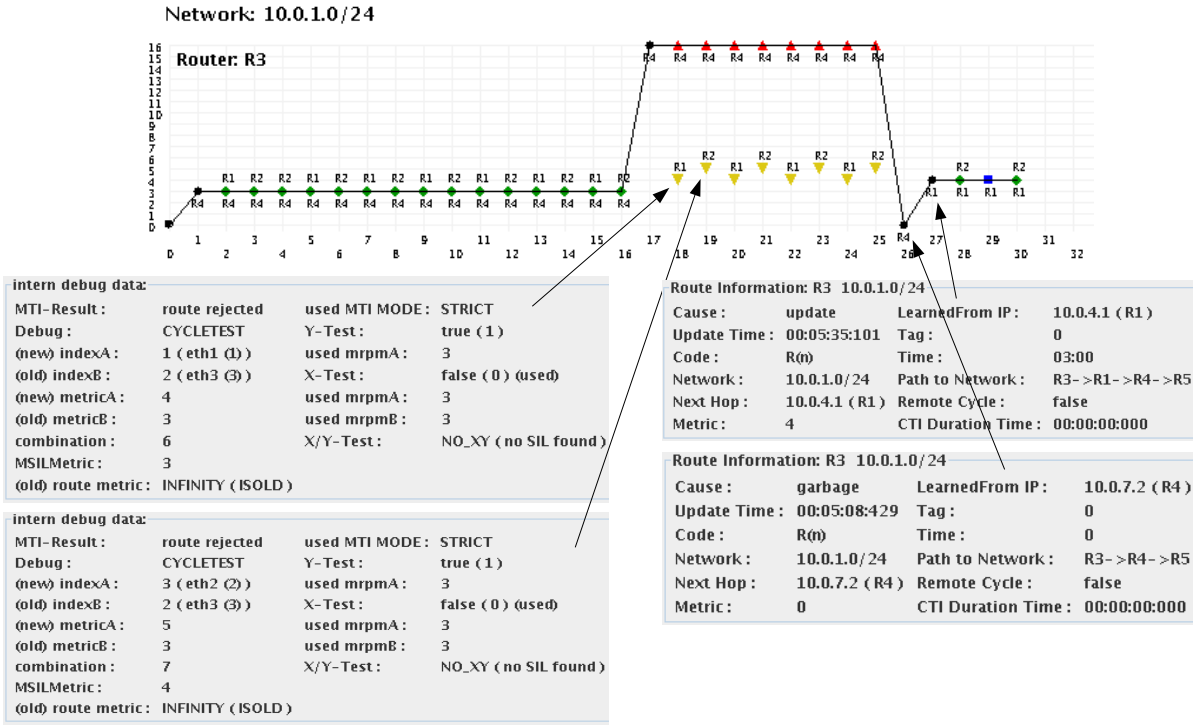


Abbildung 5.11: XTPeer Ausgabe des RIP-MTI-Strict-Mode mit aktiviertem X-Test

Wie in Kapitel 3.4.2 erläutert kann der X-Test durch den Y-Test ersetzt werden, wenn nach dem Ausfall einer Route und der Überprüfung der alternativen Route nicht auf die zuletzt gültige Route vor dem Ausfall zurückgegriffen wird, sondern auf die Route, die in einer festgelegten Zeitspanne vor dem Ausfall, von zum Beispiel zwei Update-Intervallen, die kleinste Metrik besaß. Es ist sehr wahrscheinlich, dass eine falsche alternative Route mit dieser Metrik den Source-Loop bildet. Aber auch falls das nicht der Fall sein sollte ist der verzögerte Y-Test, durch das Verwenden der kleineren Metrik, sehr viel strenger bei der Überprüfung alternativer Routen. In dem folgenden Szenario soll das Wirken des X-Tests verdeutlicht werden. In der X-Topologie in Abbildung 5.12 hat sich eine X-Kombination gebildet, die zum CTI-Problem führen wird. In Abbildung 5.13 ist der zum Verlauf der Metrik der Route von Router R1 gehörende Metrikgraph abgebildet. In R1 ist der RIP-MTI-Strict-Mode aktiv, allerdings nur mit dem „gewöhnlichen“ Y-Test. Die Route von Router R1 zum Subnetz 10.0.1.0/24 verläuft über Router R4. Router R4 fällt nun aus und R1 setzt die Metrik seiner Route zum Subnetz nach Ablauf des Timeout-Timers auf RIP-Infinity. Kurz darauf sendet Router R5 jedoch noch ein periodisches Routing-Update an R1, mit noch alten Erreichbarkeitsinformationen über das Subnetz 10.0.1.0/24. R1 übernimmt die Route von R5, die allerdings kurz darauf wieder ausfällt, da nun auch R5 den Ausfall von R4 bemerkt hat. Router R1 hatte den vorherigen Ausfall der

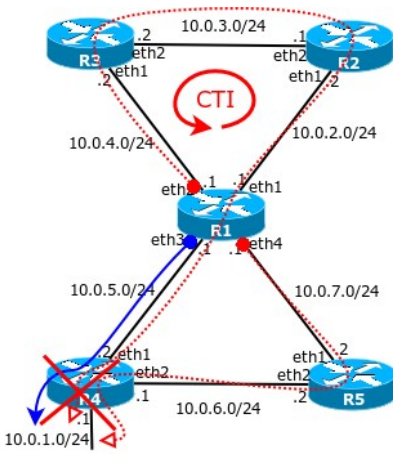


Abbildung 5.12: Die X-Kombination

Route an seine beiden Nachbarrouter mit einem Triggered-Update weitergegeben, aber nur Router R3 hat diese Information erhalten. Router R2 besitzt noch die alte Information über die Route, mit einer ungültigen Metrik, und gibt diese mit dem nächsten Routing-Update an R3 weiter. Die kurzzeitig vorhandene Information über die Route von R5 konnte R1 aufgrund der 5 Sekunden Wartezeit zwischen zwei Triggered-Updates nicht weitersenden. Die vermeintliche, alternative Route, die R1 nun von Router R3 angeboten bekommt, basiert somit nicht auf der letzten gültigen Route die R1 besaß und die über R5 führt. Der Y-Test vergleicht hier somit die falschen Routen miteinander. Aufgrund der höheren Metrik der letzten, gültigen Route erkennt er einen Simple-Loop und akzeptiert die alternative Route von R3. Der X-Test, der hier deaktiviert ist, vergleicht die MRPM-Werte mit den Routen und erkennt deswegen den Source-Loop. Auch der Y-Test würde den Source-Loop erkennen, wenn er die richtigen Routen miteinander vergleichen würde, bzw. wenn die Metrik der zuletzt gültigen Route kleiner oder gleich ist mit der Metrik der Route, auf der der Source-Loop basiert. Zur Durchführung dieses Szenarios wurde der Zeitwert der Variable „oldmetric_delay“ auf „Value = 1“ gesetzt. Die kleinere Metrik der älteren Route von R4 wird also lediglich für 1 Sekunde im RIP-MTI-Algorithmus vor dem Überschreiben durch den Eingang der neuen Route von R5 mit größerer Metrik geschützt. Ein „oldmetric_delay“ von „Update=2“, also von 2 Update-Intervallen, dürfte allerdings in allen Situationen genügen um den X-Test wirksam durch den Y-Test zu ersetzen.

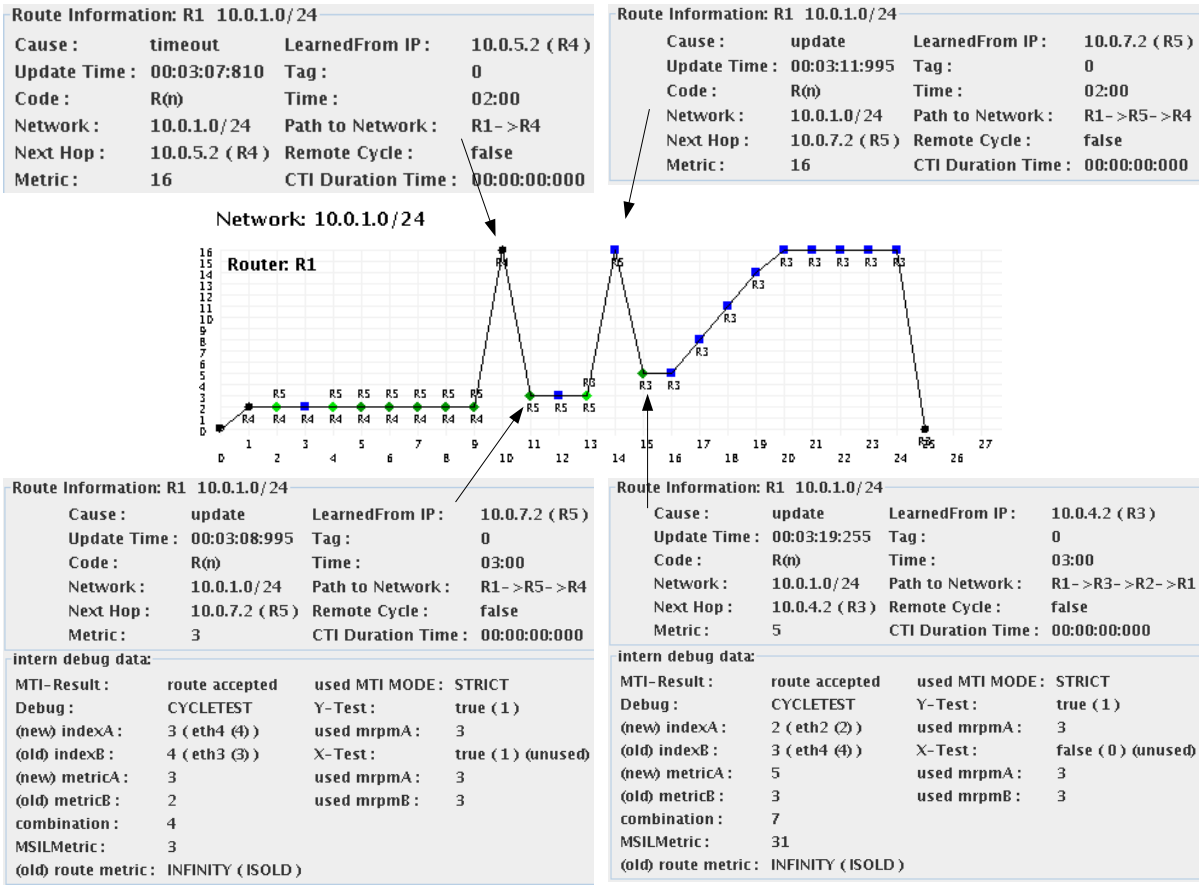


Abbildung 5.13: XTPeer Ausgabe des Y-Tests in der X-Kombination

Der verzögerte Y-Test kann allerdings auch nicht immer alle gültigen, alternativen Routen erkennen und von den ungültigen unterscheiden, insbesondere dann nicht, wenn die alternative Route nicht dem Pfad entlang verläuft, den der MRPM-Wert des Interfaces zusammen mit der Metrik der alten Route beschreibt.

In der Topologie Abbildung 5.14 fällt Router R4 komplett aus. Der RIP-MTI-Strict-Mode in Router R1 lehnt die gültige, alternative Route von Router R3 und R2 aufgrund des Y-Tests ab. Bei Router R1 liegt die gleiche Metrik wie beim CTI-Problem in Abbildung 5.8 vor. Das bedeutet letztlich, dass die Gültigkeit der Route allein an ihrer Metrik nicht zu erkennen ist.

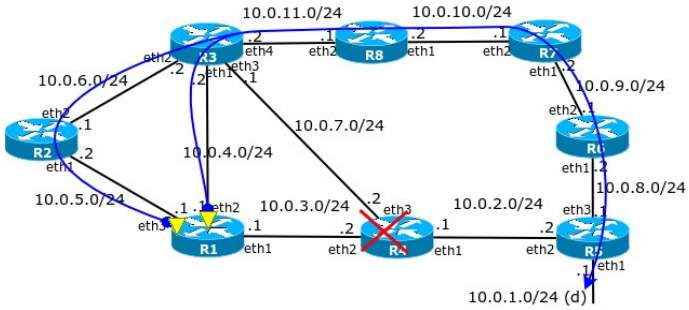


Abbildung 5.14: Der Y-Test lehnt eine gültige Route ab

Im folgenden ist der Metrikgraph des Verlaufs der Metrik der Route zum Subnetz 10.0.1.0/24 im Router R1 abgebildet. Hier zeigt sich das gleiche Bild wie in Abbildung 5.11, die alternative Route wird abgelehnt, bis die ausgefallene Route aus der Routing-Tabelle gelöscht worden ist.

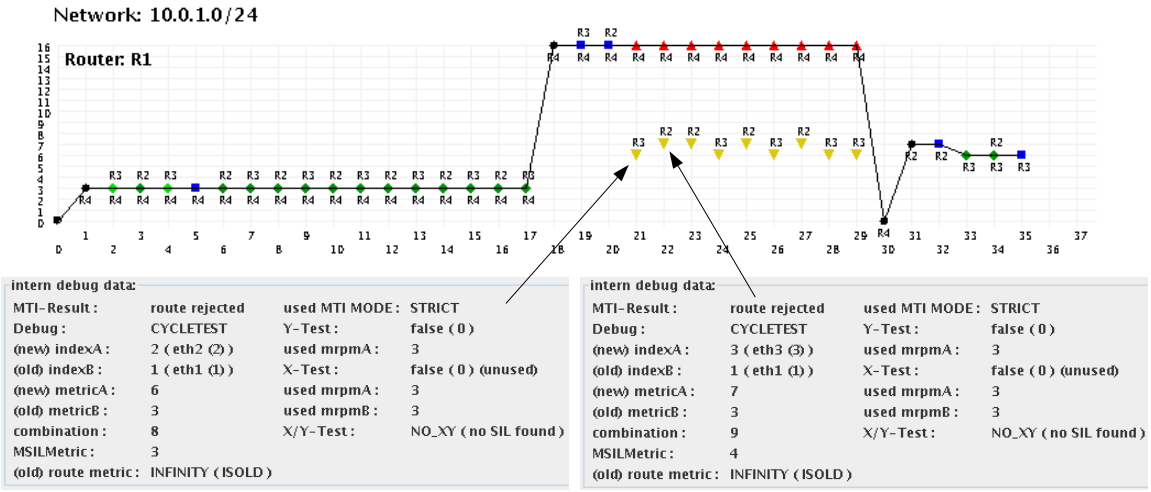


Abbildung 5.15: XTPeer Ausgabe der Ablehnung einer gültigen Route durch den Y-Test

Die Effizienz des RIP-MTI-Strict-Mode kann mit einer zeitlichen Begrenzung seiner Laufzeit erhöht werden. Verhindert der RIP-MTI-Strict-Mode die Annahme einer alternativen Route über die ganze Zeit, wie im Metrikgraphen von Abbildung 5.15 dargestellt, dann ist es sehr unwahrscheinlich, dass ein Source-Loop vorliegt, sondern sehr viel wahrscheinlicher, dass eine gültige, alternative Route vorhanden ist und diese abgelehnt wird. Es genügt daher den RIP-MTI-Strict-Mode nicht bis zum Löschen der ausgefallenen Route in der Routing-Tabelle, nach Ablauf des Garbage-Collection-Timers, aktiv zu halten, sondern vorher schon die Überprüfung dieser Route durch X-/Y-Test abzustellen und die alternative Route im Routing-Prozess weiterzugeben. Wird zudem zuvor noch ein Routing-Update, mit RIP-Infinity als Metrik für die Route, an die Nachbarrouter gesendet, so wird ein eventuell vorhandener Source-Loop mit hoher Wahrscheinlichkeit sofort gelöscht. Dies kann der RIP-MTI-Algorithmus sogar feststellen, wenn die angebotene, alternative Route nach dem Senden des Routing-Updates mit Metrik RIP-Infinity, nicht mehr mit einer gültigen Metrik, sondern nun ebenfalls mit der Metrik RIP-Infinity angeboten wird.

5.5.3 RIP-MTI-Careful-Mode

Der RIP-MTI-Careful-Mode basiert auf dem RIP-MTI-Strict-Mode und versucht dessen Effizienz zu steigern. Derzeit steht die Bezeichnung RIP-MTI-Careful-Mode mehr als Überbegriff für unterschiedliche Vorgehensweisen, den RIP-MTI-Strict-Mode einerseits abzuschwächen, um die Konvergenz des Netzwerks nach dem Ausfall einer Route wieder zu beschleunigen und mit konventionellen RIP-Routern gleichzusetzen, und andererseits die Verhinderung des CTI-Problems nach wie vor umfassend zu gewährleisten. Wie beim RIP-MTI-Strict-Mode, so wird auch beim RIP-MTI-Careful-Mode jede alternative Route, die eine ausgefallene Route in der Routing-Tabelle des Routers ersetzen soll, mit dem X-/Y-Test auf ihre Gültigkeit hin überprüft. Ist das Ergebnis des X-/Y-Tests positiv, so wird die alternative Route zugelassen, bzw. zum nächsten Entscheidungskriterium des RIP-Routing-Prozesses weitergereicht. Ist das Ergebnis negativ, so wird die alternative Route abgelehnt und nicht in die Routing-Tabelle eingetragen. Zugleich wird jedoch die ausgefallene Route, die aufgrund des RIP-MTI-Careful-Modes nun nicht ersetzt wird, markiert. Mittels der Markierung kann über drei verschiedene „Flags“ angezeigt werden, ob bisher noch keine alternative Route eingetroffen ist (no_watch), eine alternative Route eingetroffen ist aber abgelehnt wurde (watch) oder die alternative Route vom RIP-MTI-Algorithmus überprüft und für gültig befunden wurde (checked). Letzeres „Flag“ ist vor allem dann notwendig, wenn der RIP-MTI-Careful-Mode die alternative Route erst nach einer Verzögerung zulässt. Dann wird die nächste alternative Route, die über das gleiche Interface, bzw. vom gleichen Nachbarrouter aus, eintrifft, nicht mehr vom RIP-MTI-Algorithmus überprüft, sondern direkt in den Routing-Prozess weitergereicht. Die drei Flags sind in der Header-Datei „rip_mti.h“ definiert.

```
#define RIP_MTI_NO_WATCH 0    //the route is not watched by mti
#define RIP_MTI_WATCH    1    //the route was rejected by mti and is
                              // watched now until it is checked as ok
#define RIP_MTI_CHECKED  2    //The route is checked by mti
```

Mit Hilfe der Markierung in der ausgefallenen Route kann somit erkannt werden, ob bereits zuvor eine alternative Route eingetroffen ist, die abgelehnt wurde. Es ist höchst unwahrscheinlich das ein Source-Loop eine längere Zeit übersteht, da er generell sehr instabil ist und durch ein einziges Routing-Update mit Metrik RIP-Infinity auch wieder gelöscht werden kann. Das erlaubt dem RIP-MTI-Careful-Mode die alternative Route nach dem obligatorischen X-/Y-Test auch mit weiteren Kriterien zu überprüfen, die weniger streng und unabhängig von den gegenwärtigen Metriken der Routen und Simple-Loops sind. Die verschiedenen Varianten des RIP-MTI-Careful-Mode unterscheiden sich darin, dass sie bei Ablehnung der alternativen Route durch den X-/Y-Test unterschiedliche Informationen zurückhalten und mit der ausgefallenen Route verknüpfen. Während beim RIP-MTI-Normal-Mode und RIP-MTI-Strict-Mode die Überprüfung der alternativen Route nach dem Verlassen des RIP-MTI-Algorithmus abgeschlossen ist und mit wiederholtem eintreffen der gleichen alternativen Route eine komplett neue Überprüfung beginnt, werden beim RIP-MTI-Careful-Mode, bei negativem Ergebnis des X-/Y-Tests, zusätzliche Prozesse angestoßen, die von weiteren angebotenen, alternativen Routen beeinflusst werden und bis zum Ersatz oder der Bestätigung der ausgefallenen Route aus der Routing-Tabelle, unter Umständen auch bis zu ihrer Löschung, andauern können.
Im folgenden werden nun die bisher implementierten Varianten des RIP-MTI-Careful-Mode vorgestellt, die das CTI-Problem einerseits immer verhindern, aber andererseits die Konvergenz des Netzwerks nicht verlangsamen sollen.

5.5.3.1 RIP-MTI-Careful-ESHC-Mode

Die Abkürzung ESHC beim RIP-MTI-Careful-ESHC-Mode steht für External-Split-Horizon-Check. Der RIP-MTI-Careful-ESHC-Mode überprüft nach dem ersten Angebot einer alternativen Route, die eine ausgefallene Route ersetzen soll, aber vom X-/Y-Test abgelehnt wurde, ob ihm jeweils eine weitere alternative Route zum gleichen Subnetz über alle Interfaces angeboten wird (bzw. von allen Nachbarroutern), die über einen Simple-Loop mit dem Interface der zuerst angebotenen alternativen Route in Verbindung stehen. Wenn ein Source-Loop vorliegt dann hat einer der Nachbarrouter, aus der angrenzenden topologischen Netzwerkschleife, die Route vom Router übernommen und den Split-Horizon bzgl. der Route zum Router hin gesetzt, weswegen er die Route auch nicht mehr dem Router anbieten wird. Ist die alternative Route dagegen gültig und liegt kein Source-Loop vor, so dürfte keiner der Nachbarrouter den Split-Horizon bzgl. der Route zum Router gesetzt haben, da die Route von „Außerhalb“ der Netzwerkschleife kommt.
Mit dem Ablehnen der alternativen Route durch den X/Y-Test wird die ausgefallene Route in der Routing-Tabelle nicht ersetzt. Der RIP-MTI-Careful-ESHC-Mode sucht sich nun aus der MSILM-Tabelle alle Interfaces, die mit dem Interface der abgelehnten alternativen Route einen Simple-Loop haben, heraus und speichert sie in der ESHC-Liste, die mit der ausgefallenen Route aus der Routing-Tabelle verknüpft ist. Die ESHC-Liste ist ein Vektor, in den schlichtweg Referenzen (Pointer) auf die entsprechenden Einträge aus der MSILM-Tabelle gespeichert werden. Die Referenzen auf die MSILM-Einträge werden an den Positionen in der ESHC-Liste gespeichert, an denen die Interfaces in der MRPM-Tabelle stehen. Handelt es sich um eine gültige alternative Route, die auf ein erreichbares Subnetz verweist, dann wird sie sich im benachbarten Umfeld des Routers verteilen und über alle zusammenhängende Interfaces angeboten werden. Mit jedem Angebot einer alternativen Route über ein Interface wird der zugehörige Eintrag aus der ESHC-Liste gelöscht. Um nicht auf Interfaces zu warten, über die keine alternative Route eintreffen kann, weil der Simple-Loop ausgefallen ist, wird über die Referenz auf den MSILM-Eintrag auch die Zeit der letzten Bestätigung des MSILM-Wertes kontrolliert. Liegt die letzte Bestätigung eines MSILM-Wertes zu lange zurück, wird der Eintrag der Referenz ebenfalls aus der ESHC-Liste gelöscht. Mit dem Lösche45n des letzten Eintrags aus der ESHC-Liste wird die Route angenommen. Handelt es sich bei der Route um einen Source-Loop, dann wird mindestens ein Eintrag in der ESHC-Liste stehen bleiben, aufgrund des Nachbarrouters der wegen des gesetzten Split-Horizon die Route dem Router nicht anbietet. Der RIP-MTI-Careful-ESHC-Mode wurde als aktiver und als passiver Modus implementiert. Beim aktiven Modus werden bzgl. der Interfaces aus der ESHC-Liste Response- und Request-Nachrichten über die Interfaces an die Nachbarrouter gesendet, um einen eventuell vorhandenen Source-Loop gleich zu löschen, bzw. um die Leerung der ESHC-Liste zu beschleunigen. Bei den bisherigen praktischen Tests in der Testumgebung mit dem XTPeer hat sich das bislang aber als nicht sonderlich beschleunigend herausgestellt, da RIP bei Veränderungen ohnehin Triggered-Updates verwendet.
In Abbildung 5.16 ist die Situation dargestellt. Router R1 lehnt die alternative Route zum Subnetz 10.0.1.0/24 aufgrund des X-/Y-Tests ab. Liegt eine Situation wie in Abbildung 5.14 vor, dann bekäme Router R1 auch über Interface eth3 eine alternative Route angeboten, was

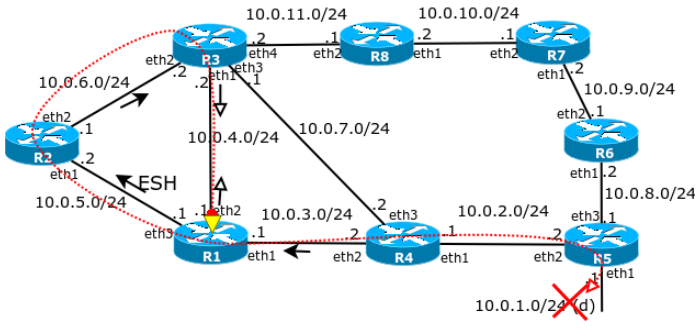


Abbildung 5.16: Source-Loops durch Careful-ESHC erkennen

hier aufgrund des External-Split-Horizon von Router R2 nicht der Fall ist. Welche Interfaces über einen Simple-Loop miteinander verfügen kann über die MSILM-Tabelle erkannt werden.
Im folgenden ist der Metrikgraph des Verlaufs der Metrik der Route von Router R1 zum Subnetz 10.0.1.0/24 abgebildet. Der Metrikgraph beschreibt die Situation aus Abbildung 5.14, also die Existenz einer gültigen, alternativen Route über einen längern Pfad, der nicht über einen erkannten Simple-Loop erfasst ist. Die erste Ablehnung der alternativen Route geschieht durch den X-/Y-Test (X-Test ist abgeschaltet; „unused“). In die „ESHC-List“ wird dabei gleichzeitig der Positionsindex des Eintrags aus der MRPM-Tabelle eingetragen, der das Interface eth3, hinter dem Router R2 zu finden ist, enthält. Das Interface eth1, mit dem eth2 ebenfalls einen Simple-Loop besitzt, wird nicht eingetragen, da der Simple-Loop schon seit längerer Zeit nicht mehr bestätigt wurde (hier 204 Sekunden). Dies kann aus der MSILM-Tabelle über den Eintrag eth1/eth2 gelesen werden. Trifft die alternative Route auch von Router R2 über eth3 beim Router ein, wird der Eintrag aus der ESHC-Liste gelöscht. Da die ESHC-Liste anschließend leer ist kann hier kein Source-Loop vorliegen und die Route müsste gültig sein.

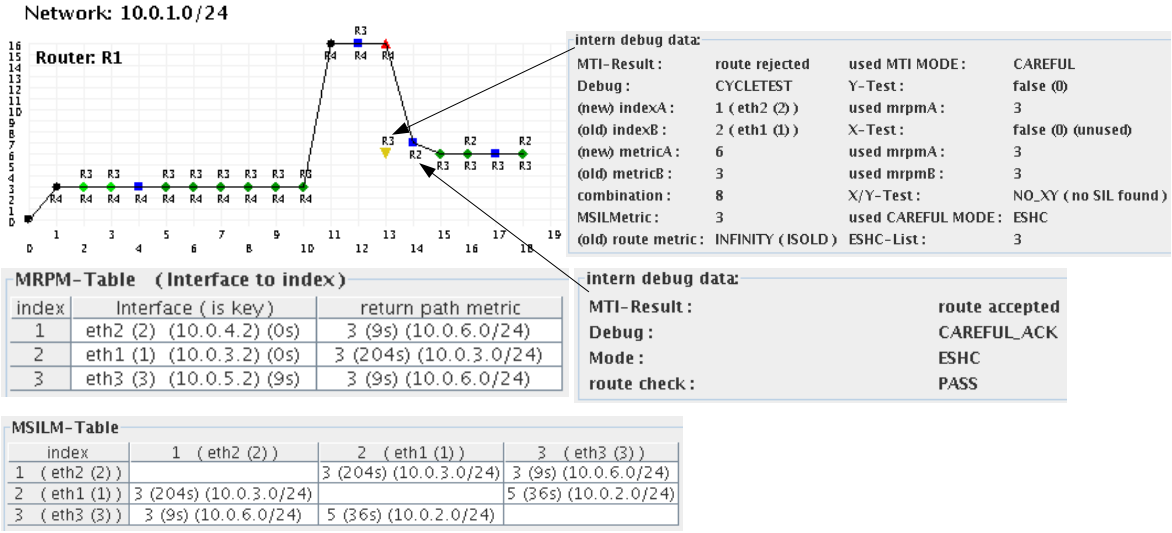


Abbildung 5.17: XTPeer Ausgabe des RIP-MTI-Careful-ESHC-Mode

Der Vorteil des RIP-MTI-Careful-ESHC-Mode ist, dass er unabhängig von internen Timern arbeitet und eine gültige, alternative Route so früh wie möglich akzeptiert.
Von Nachteil ist jedoch, dass die alternative Route zum Subnetz, deren Interface als letztes aus der ESHC-Liste gelöscht wird, auch angenommen wird. Diese letzte Route ist aber meist auch die mit der schlechtesten Metrik, weswegen es meist kurz nach der Annahme der Route erneut zu einer Veränderung in der Routing-Tabelle kommt. Dies ist auch im Metrikgraphen in Abbildung 5.17 zu sehen. Die alternative Route von R2 wird zuerst angenommen und erst danach wird die alternative Route über R3, mit der kleineren Metrik in die Routing-Tabelle übernommen. Außerdem ist im Zusammenhang mit dem RIP-MTI-Careful-Mode noch offen, ob es zu Situationen kommen kann, in denen sich zwei Router gegenseitig blockieren, weil sie jeweils auf das Eintreffen der alternativen Route über die entsprechenden Interfaces warten, bevor sie selbst die Route annehmen und weitergeben.
Ein weiterer Nachteil des RIP-MTI-Careful-ESHC-Mode ist, dass er von einer vollständigen MSILM-Tabelle abhängig ist, wie in Kapitel 3.4.3 beschrieben, sonst existieren „Schlupflöcher“ für das CTI-Problem. Eine vollständige MSILM-Tabelle kann zwar errechnet werden, dieser Vorgang erhöht jedoch auch den Aufwand, den der Router für den RIP-MTI-Algorithmus aufbringen muss. Das Problem der Unvollständigkeit der MSILM-Tabelle wurde erst mit dem RIP-MTI-Careful-ESHC-Mode festgestellt.

Letztlich muss auch der Mehraufwand, den diese Technik verursachtet, berücksichtigt werden. Für jede ausgefallene Route in der Routing-Tabelle wird eine ESHC-Liste mit entsprechenden Überprüfungen geführt. Sollten aufgrund eines größeren Schadens im Netzwerk hunderte von Routen ausfallen, so könnte das zu einer erheblichen Mehrbelastung führen, die das Routersystem aushalten können müsste.
Aufgrund dieser vielen Nachteile kann der RIP-MTI-Careful-ESHC-Mode letztlich nicht zur Verwendung empfohlen werden.

5.5.3.2 RIP-MTI-Careful-DT-Mode

Die Abkürzung DT beim RIP-MTI-Careful-DT-Mode steht für Deny Timer. Der RIP-MTI-Careful-DT-Mode lehnt eine gültige, alternative Route nur für eine relativ kurze Zeit ab, wenn sie den X-/Y-Test nicht besteht. In dieser Zeit versucht er einen eventuell bestehenden Source-Loop in den Routern der angrenzenden topologischen Netzwerkschleife durch das Senden eines Routing-Updates, mit Metrik RIP-Infinity, zu löschen. Eine gültige, alternative Route kann dagegen auf diese Weise nicht gelöscht werden. Die zuvor abgelehnte, alternative Route wird im Router gespeichert und nach Ablauf der Zeitspanne vom RIP-MTI-Careful-DT-Mode selbst wieder in den Routing-Prozess eingesetzt. Wird während der Zeitspanne der Ablehnung statt der vormals alternativen Route nun eine Route mit Metrik RIP-Infinity über das gleiche Interface angeboten, so wird die zwischengespeicherte Route verworfen und die ausgefallene Route in der Routing-Tabelle bleibt ausgefallen.
Mit dem Ablehnen der Route durch den X/Y-Test wird die Annahme der alternativen Route zum Subnetz blockiert. Diese abgelehnte, alternative Route wird lokal im RIP-MTI-Algorithmus zwischengespeichert und es wird ein Thread von bestimmter Dauer gestartet. Gleichzeitig werden über alle Interfaces, die mit dem Interface der abgelehnten Route über einen Simple-Loop in Verbindung stehen, ein Routing-Update mit der Information über den Ausfall der Route gesendet. Liegt ein Source-Loop vor, dann wird dieses Routing-Update binnen kürzester Zeit durch die angrenzende topologische Netzwerkschleife wandern, die falsche alternative Route in den dortigen Routern „überschreiben“ und dem Router über das gleiche Interface angeboten, wie zuvor die falsche alternative Route. Die miteinander Verbundenen Interfaces können über die MSILM-Tabelle gefunden werden. Trifft also über das gleiche Interface nun eine Route zum Subnetz mit Metrik RIP-Infinity ein, so wird der Thread gestoppt und die gespeicherte Route wird gelöscht. Läuft der Thread dagegen aus, dann wurde das Routing-Update mit der Information über den Ausfall der Route nicht durch die Netzwerkschleife geleitet, was bedeutet, dass die angebotene, alternative Route gültig sein müßte. Die zuvor abgelehnte aber zwischengespeicherte Route wird dann nachträglich in die Routing-Tabelle eingetragen. Dafür wird die ausgefallene und noch nicht ersetzte Route in der Routing-Tabelle mit dem „checked“-Flag markiert und die zuvor abgelehnte Route wird nachträglich wieder in den Routing-Prozess eingesetzt. Ansonsten könnte nun aber auch jede beliebige andere Route zum Subnetz, die über das Interface eintrifft angenommen werden.
In Abbildung 5.18 ist die Situation dargestellt, wie der RIP-MTI-Careful-DT-Mode den Source-Loop durch das Aussenden eines Routing-Updates, mit

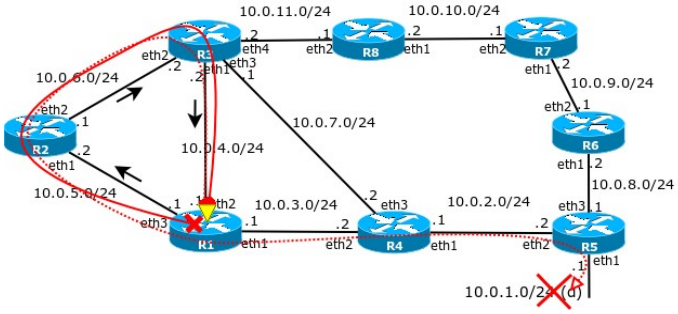


Abbildung 5.18: Source-Loops durch den Careful-DT erkennen

der Metrik RIP-Infinity, bzgl. der ausgefallenen Route zum Subnetz 10.0.1.0/24 in der Netzwerkschleife über den Router R1, R2 und R3 „überschreibt“. In einer Situation, wie sie in Abbildung 5.14 beschrieben ist, kommt das ausgesendete Routing-Update nicht durch die Netzwerkschleife und die alternative Route bleibt bestehen und wird weiterhin angeboten. Welche Zeit der RIP-MTI-Careful-DT-Mode warten sollte, bis er die zurückgehaltene Route annehmen kann ist allerdings noch offen. Möglich wäre eine Zeitspanne die Abhängig vom Update-Intervall oder aber auch von der Metrik der erkannten zugehörigen Simple-Loops ist.
Im folgenden ist der Metrikgraph des Verlaufs der Metrik der Route von Router R1 zum Subnetz 10.0.1.0/24 abgebildet. Der Metrikgraph beschreibt die Situation aus Abbildung 5.14 und damit die Existenz einer gültigen, alternativen Route und wie der RIP-MTI-Careful-DT-Mode damit umgeht. Mit der Ablehnung der gültigen, alternativen Route durch den X-/Y-Test wird ein Thread für 30 Sekunden gestartet, was der Zeit eines Update-Intervalls entspricht. Es kann anhand der Update-Time (00:03:54:261) erkannt werden das ca. 30 Sekunden später erneut ein Routing-Update von R3 eintrifft, welches allerdings noch abgelehnt wird. Die Metrik und die Learned-From-IP-Adresse werden hier überprüft und gegebenenfalls würde die zwischengespeicherte Route aktualisiert werden. Nur hundertstel Sekunden (00:03:54:443) nach diesem Routing-Update trifft ein weiteres ein, das nun übernommen wird. An dieser Stelle ist nun der Thread abgelaufen und der RIP-MTI-Careful-DT-Mode trägt die Route, die er zwischengespeichert hatte, nun selbst in den Routing-Prozess ein. Letztlich wird sie dann auch in die Routing-Tabelle eingetragen, als Ersatz für die ausgefallene Route.

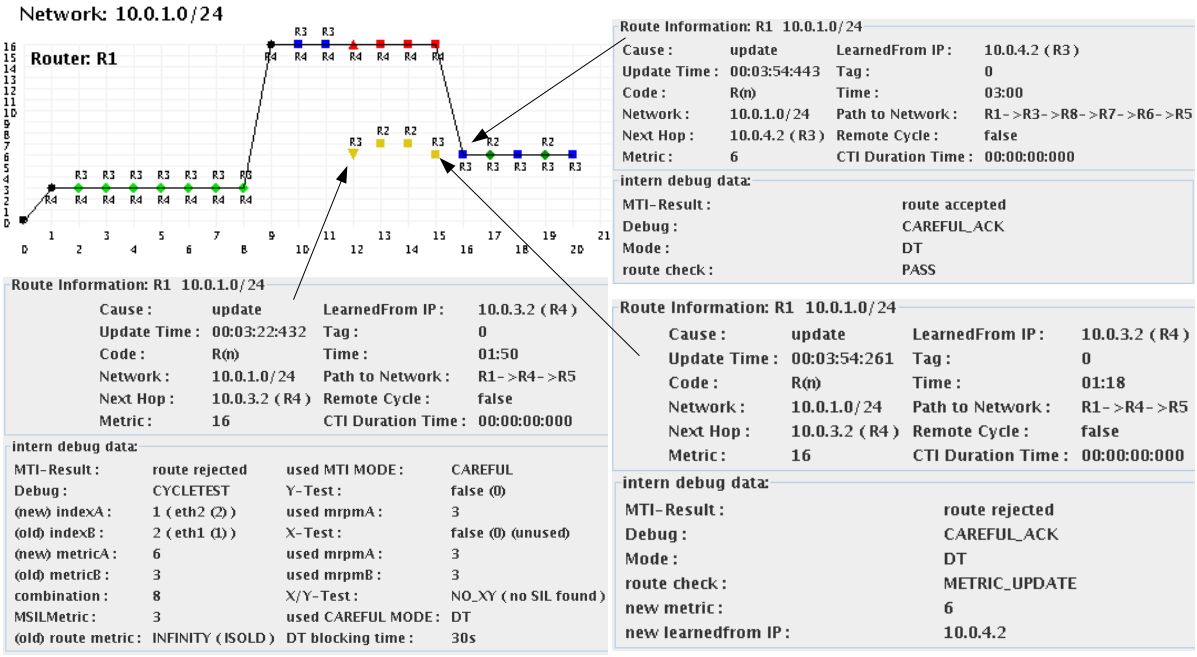


Abbildung 5.19: XTPeer Ausgabe des RIP-MTI-Careful-DT-Mode

Vorteil des RIP-MTI-Careful-DT-Mode ist, dass er die sich als gültig erweisende Route sofort einsetzen kann, da er sie zwischenspeichert. Bei einer gut gewählten Dauer des Threads ist die zeitliche Verzögerung minimal, so dass der RIP-MTI-Careful-DT-Mode ein guter Kompromiss zwischen der Verhinderung des CTI-Problems und der Beeinträchtigung des Konvergenzverhaltens des Netzwerks darstellt.
Nachteil des RIP-MTI-Careful-DT-Mode ist, dass er von einem Rundlauf des in die angrenzende Netzwerkschleife ausgesendeten Routing-Updates abhängig ist und noch offen ist, ob dieser Rundlauf wirklich nur aufgrund einer vorhandenen, gültigen, alternativen Route scheitern

kann. Nach dem Ausfall einer Route und der „Neu-Ausrichtung“ des Netzwerks bzgl. der Erreichbarkeit des Subnetzes, kann es auch dazu kommen, dass dem Router über zeitlich ungünstig gesendete, periodische Routing-Updates einmalig alternative Routen mit nun nicht mehr gültigen Metriken angeboten werden. Diese Routen könnten durch den RIP-MTI-Careful-DT-Mode zwischengespeichert und nach Ablauf der Zeitspanne und Beendigung des Threads in die Routing-Tabelle des Routers eingesetzt werden. Damit dies nicht passiert, muss die Metrik der zwischengespeicherten, alternativen Route, wie in Abbildung 5.19 zu sehen, mit jedem eingehenden Routing-Update überprüft und gegebenenfalls aktualisiert werden.
Zu beachten ist auch die Mehrbelastung für den Router, die vom RIP-MTI-Careful-Mode letztlich ausgeht. Für jede ausgefallene Route in der Routing-Tabelle, für die eine alternative Route abgelehnt wurde, muss ein Thread gestartet werden und zudem die abgelehnte Route im Router zwischengespeichert werden.
Aufgrund der Nachteile des RIP-MTI-Careful-DT-Mode wird er letztlich nicht zur Anwendung empfohlen. Von ihm wurde allerdings die nächste und letzte hier vorgestellte Variante des RIP-MTI-Careful-Mode abgeleitet.

5.5.3.3 RIP-MTI-Careful-RT-Mode

Die Abkürzung RT beim RIP-MTI-Careful-RT-Mode steht für Request Timer. Der RIP-MTI-Careful-RT-Mode funktioniert im wesentlichen genau wie der RIP-MTI-Careful-DT-Mode, mit dem Unterschied, dass die abgelehnte, alternative Route nicht lokal gespeichert wird. Stattdessen wird nach Ablauf der Zeitspanne, bzw. des Threads, eine Request-Nachricht ausgesendet, um von den Nachbarroutern den gegenwärtigen Status einer Route zum nicht mehr erreichbaren Subnetz zu erfahren. Der RIP-MTI-Careful-RT-Mode wurde so implementiert, dass er unabhängig von der MSILM-Tabelle funktioniert, falls auf diese verzichtet werden würde, da der RIP-MTI-Strict-Mode sie letztlich auch nicht benötigt.
Wenn der X-/Y-Test eine gültige, alternative Route ablehnt wird ein Thread von bestimmter aber kurzer Dauer gestartet. Ein Routing-Update mit Metrik RIP-Infinity wird in die angrenzende Netzwerkschleife gesendet, um einen eventuell vorhandenen Source-Loop zu löschen. In der Zeit in der der Thread läuft wird jede weitere eingehende, alternative Route zum Subnetz direkt abgelehnt. Nach Ablauf des Threads wird eine RIP-Request-Nachricht an die Nachbarrouter gesendet. Wie beim RIP-MTI-Careful-DT-Mode auch, wird der Thread über ein eingehendes Routing-Update mit der Metrik RIP-Infinity vor Ablauf der Zeit gestoppt und die Request-Nachricht wird nicht gesendet. Wie in Abbildung 5.18 dargestellt bedeutet das auch hier, dass der Source-Loop „überschrieben“ werden konnte.
Im folgenden Metrikgraphen in Abbildung 5.20 ist der Verlauf der Metrik der Route vom Router R1 zum Subnetz 10.0.1.0/24 abgebildet mit der Situation aus Abbildung 5.14, also dem Vorhandensein einer gültigen, alternativen Route und der Reaktion des RIP-MTI-Careful-RT-Mode darauf. Der Unterschied zum RIP-MTI-Careful-DT-Mode ist der, dass am Ende des Threads die neue Route über ein RIP-Request von den Nachbarroutern eingeholt wird. Mit der Ablehnung der gültigen, alternativen Route durch den X-/Y-Test wird hier der Thread für 30 Sekunden gestartet, was hier der Update-Intervall-Zeit entspricht. Ein letztes Routing-Update wird kurz vor Ablauf des Threads nochmal abgelehnt (00:03:45:432). Würde im Anschluss des Threads kein Request gesendet, so müsste der Router in diesem Fall weitere 30 Sekunden auf das nächste periodische Routing-Update warten, bevor er eine alternative Route zum Subnetz erhält. Die RIP-Request-Nachricht beschleunigt daher die Konvergenz des Netzwerks nach dem Ausfall einer Route. Da nach Ablauf des Threads und vor dem Senden des Requests die

ausgefallene Route in der Routing-Tabelle mit dem Flag „checked“ markiert wird, überprüft der RIP-MTI-Careful-RT-Mode die nachfolgend, eintreffenden Routen zum Subnetz nicht mehr.

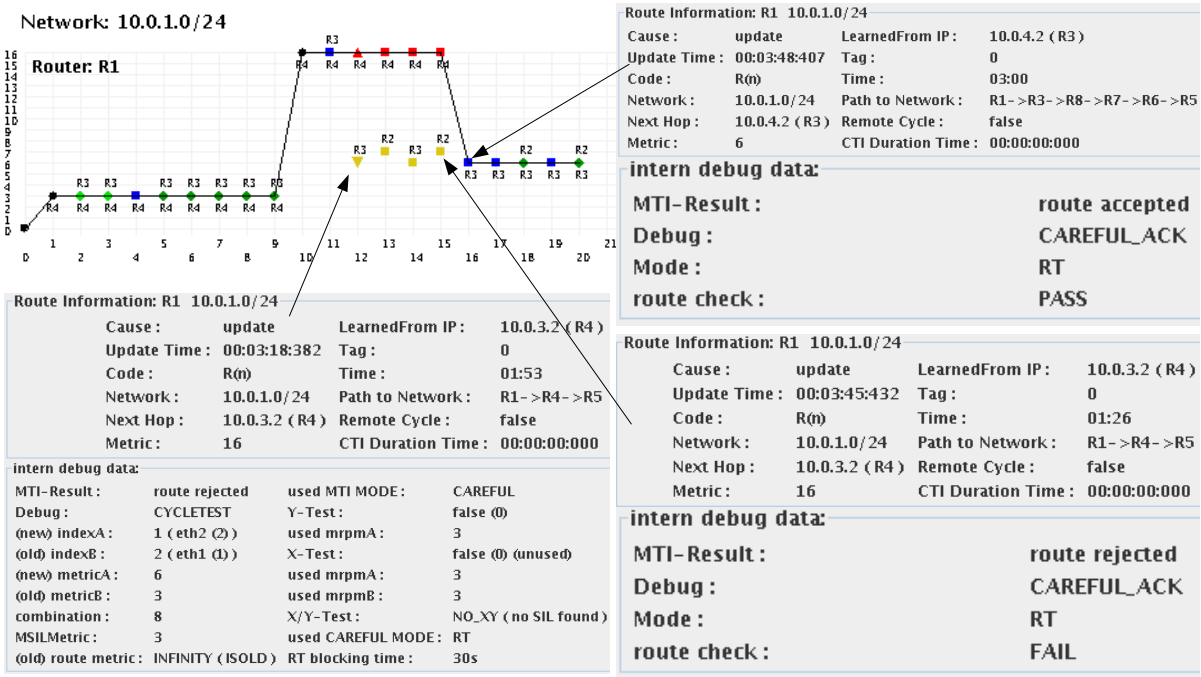


Abbildung 5.20: XTPeer Ausgabe des RIP-MTI-Careful-RT-Mode

Der RIP-MTI-Careful-RT-Mode ist letztlich die hier empfohlene Variante einer Implementierung des RIP-MTI-Careful-Mode. Er ist zwar vom Prinzip her dem RIP-MTI-Careful-DT-Mode zu vergleichen, benötigt aber weit weniger Ressourcen, da er keine Route lokal zwischenspeichert. Mit einer kurzen Zeit für die Dauer des Threads, verzögert er die Konvergenz des Netzwerks nach dem Ausfall einer Route nur gering. Die Anzahl der zu sendenden Requests kann gering gehalten werden, wenn nicht für jede zu überprüfende Route ein entsprechender Thread gestartet wird, sondern für jedes Interface, über das dann nach Ablauf der Zeit ein Request gesendet werden soll. Allerdings sind noch einige Fragen bzgl. des Sendens der RIP-Request-Nachrichten offen. Für gewöhnlich sendet ein RIP-Router RIP-Request-Nachrichten nur einmal direkt nach dem Start des Routers, um gleich alle Routen des Netzwerks von seinen Nachbarroutern zu erhalten. Ansonsten erlaubt die RIP-Spezifikation auch Analysewerkzeugen das Abfragen der Routen eines Routers mittels Requests. Hier ist die Frage offen, ob über Requests in allen RIP-Implementierungen die Split-Horizon Regel eingehalten wird. Nach der RIP-Spezifikation muss Split-Horizon eingehalten werden, wenn alle Routen abgefragt werden, aber nicht wenn gezielt eine Route abgefragt wird. Einer Request-Nachricht kann ein spezifisches Subnetz übergeben werden, welches dann gezielt abgefragt wird, wogegen das leere Request die komplette Routing-Tabelle abfragt. Der RIP-MTI-Careful-RT-Mode ist derzeit so eingestellt, dass wenn die Variable Triggered-Interval auf 0 steht, also Request-Nachrichten nicht gebündelt werden, Requests gesendet werden mit denen gezielt nach dem entsprechenden Subnetz der überprüften Route gefragt wird. Wird ein Triggered-Interval gesetzt, also die Request-Nachrichten nicht sofort, sondern immer erst nach Ablauf des Triggered-Intervalls gebündelt gesendet, dann wird die komplette Routing-Tabelle des Nachbarrouters per leerem RIP-Request abgefragt. Empfohlen wird allerdings letztere Einstellung. Diese Erzeugt zwar mehr Datenverkehr, dürfte allerdings auch sicherer beim Umgang mit dem CTI-Problem sein.

5.6 Loop Metric Increase Deny Timer (LMIDT)

Der LMIDT baut im Gegensatz zu den RIP-MTI-Modes auf einem anderen Entscheidungskriterium auf, um das CTI-Problem zu unterbinden. Während die RIP-MTI-Modes gegenwärtig nur über unterschiedliche Interfaces eingehende Routen zu einem Subnetz überprüfen, also nur zueinander alternative Routen, überprüft der LMIDT alle Routen, die über das gleiche Interface, vom gleichen Router kommen darauf, ob ihre Metrik um die Metrik eines minimalen Simple-Loops des zugehörigen Interface gewachsen ist. Ist das der Fall, könnte dies das Ergebnis des CTI-Problems sein. Der LMIDT versucht somit bereits entstandene Routing-Loops anhand eines ablaufenden CTI-Problems zu erkennen und zu unterbrechen. Ansonsten arbeitet der implementierte LMIDT weitestgehend wie der RIP-MTI-Careful-DT-Mode. Der LMIDT ist ein einfacher Entwurf um die Schwäche der gegenwärtigen RIP-MTI-Modes zu beheben, dass ihre Wirksamkeit von ihrer Lokalität im Netzwerk abhängig ist. Das CTI-Problem kann von einem RIP-MTI-Router bisher nur verhindert werden, bevor es entstanden ist und bevor ein Source-Loop zum Routing-Loop geworden ist. Gegenüber einem bereits vorhandenen Routing-Loop sind die gegenwärtigen RIP-MTI-Modes machtlos und ein damit verbundenes CTI-Problem kann von ihnen weder erkannt noch unterbrochen werden. Ist ein konventioneller RIP-Router an der Stelle im Netzwerk, an der ein Source-Loop zu einem Routing-Loop wird, was dieser nicht verhindern kann, so kann auch ein unmittelbar benachbarter RIP-MTI-Router dem entstandenen Routing-Loop und einem damit verbundenem CTI-Problem nicht mehr entgegentreten, da sich von nun an bei ihm alles über ein einziges Interface abspielt, was vom RIP-MTI-Algorithmus nicht berücksichtigt wird.

Der LMIDT überprüft die eingehenden Routen zu einem Subnetz, die über das gleiche Interface, bzw. vom gleichen Nachbarrouter, wie die vorhandene Route aus der Routing-Tabelle angeboten werden und um die Metrik eines gespeicherten Simple-Loops des zugehörigen Interfaces aus der MSILM-Tabelle gewachsen sind. Die Route in der Routing-Tabelle ist zwar nicht ausgefallen, würde allerdings durch die neue Route ersetzt werden, da beide vom gleichen Nachbarrouter stammen. Ist die Metrik der neuen Route um die Metrik eines Simple-Loops des zugehörigen Interfaces gewachsen, so könnte dies das Ergebnis des CTI-Problems sein. Der LMIDT lehnt dann die neue Route ab, speichert sie aber lokal im Router zwischen. Er setzt die Metrik der bislang noch gültigen Route in der Routing-Tabelle auf RIP-Infinity und sendet ein Routing-Update über alle Interfaces, die mit dem zugehörigen Interface einen Simple-Loop besitzen, um den eventuell vorhandenen Routing-Loop zu überschreiben. Desweiteren wird ein Thread mit einer bestimmten Laufzeit gestartet. Trifft erneut eine Route über das gleiche Interface, aber diesmal mit Metrik RIP-Infinity ein, so wird der Thread gestoppt, die lokal zwischengespeicherte Route wird verworfen und bleibt abgelehnt. Läuft der Thread jedoch ab, weil keine Route zum gleichen Subnetz mit Metrik RIP-Infinity eingetroffen ist, so wird die abgelehnte und zwischengespeicherte Route an den Routing-Prozess weitergereicht um in die Routing-Tabelle eingetragen zu werden.

In Abbildung 5.21 lehnt der LMIDT in R2 die Route zum Subnetz 10.0.1.0/24 über sein Interface eth1 ab und erkennt in diesem Fall das vorhandene CTI-Problem. Die Metrik der Route ist um die Metrik des Simple-Loops zwischen den Interfaces eth1 und eth2 gewachsen. Der LMIDT setzt die Metrik der Route in der Routing-

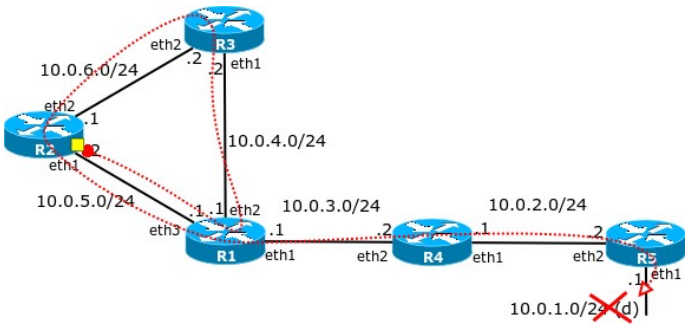


Abbildung 5.21: Der LMIDT in der Y-Topologie

Tabelle auf RIP-Infinity (16) und blockiert, hier für 15 Sekunden, alle weiteren Routen zum Subnetz 10.0.1.0/24. Mit Ablauf der Zeit würde der LMIDT die abgelehnte Route nachträglich in die Routing-Tabelle eintragen. Die 15 Sekunden beruhen in diesem Beispiel auf einem Timer der hier mit MAXMSILM eingestellt ist (siehe Kapitel 4.3.1). In Abbildung 5.22 wird der Thread über das eingehende Routing-Update mit Metrik RIP-Infinity (16) für die Route zum Subnetz 10.0.1.0/24 über Interface eth1 gestoppt. Die abgelehnte Route, die im Router zwischengespeichert wurde, bleibt abgelehnt und wird verworfen.

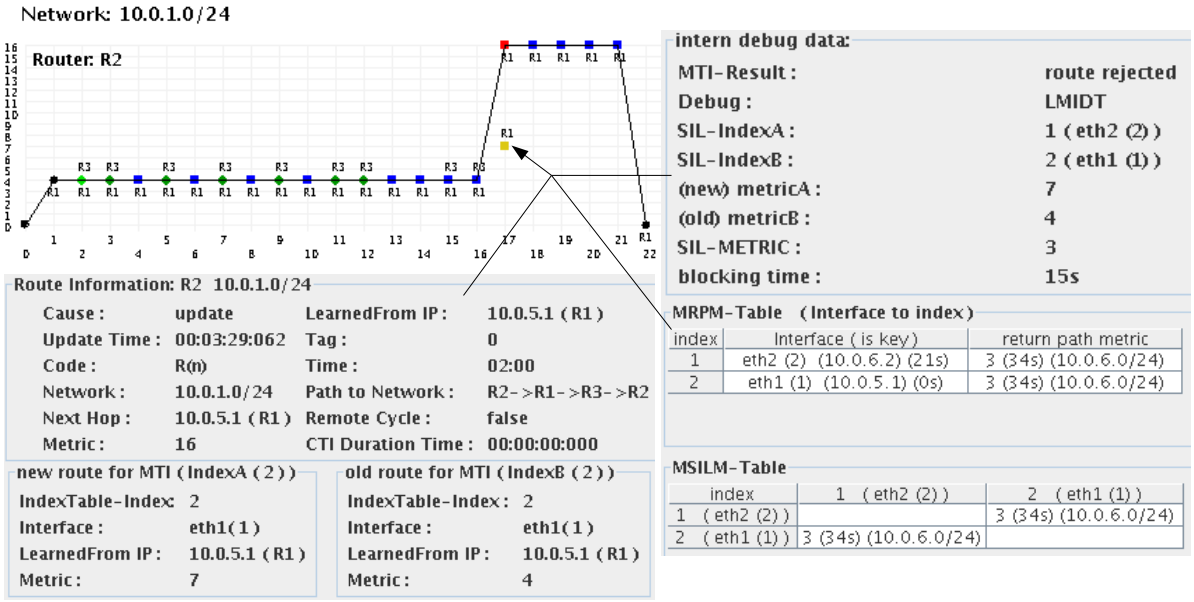


Abbildung 5.22: XTPeer Ausgabe des LMIDT

Die Vorgehensweise des LMIDT hat allerdings auch den Nachteil, dass das CTI-Problem nur unterbrochen werden kann und der Routing-Loop bereits entstanden ist, im Gegensatz zu den RIP-MTI-Modes die mit dem Source-Loop bereits das Entstehen des Routing-Loops verhindern.
Da die Implementierung des LMIDT vom RIP-MTI-Careful-DT-Mode abgeleitet wurde, verursacht er ebenfalls eine große Mehrbelastung im Routersystem, wenn viele Routen ausfallen. Hinzu kommt allerdings noch das Problem, dass wenn die Erhöhung der Metrik der Route gültig ist und kein Routing-Loop vorliegt, sich die Konvergenz des Netzwerks erheblich verlangsamen kann zumal hier die Gefahr größer ist, dass dann mehrere LMIDT Router, die in der gleichen Netzwerkschleife liegen, die Route ablehnen, während bei den RIP-MTI-Modes, der Router auf dem sie aktiviert sind und eine Route blockieren, der Punkt im Netzwerk ist von dem an die neue, alternative Route zumindest den gleichen Weg entlang läuft wie die vorherige Route und die nachfolgenden Router allenfalls eine Veränderung in der Metrik bemerken.

5.7 Orientierung über die Learned-From-IP-Adresse

In dieser Ausarbeitung wurde stets an der Orientierung des RIP-MTI-Algorithmus über die Interfaces des Routers festgehalten, doch nur wegen der erhofften höheren Verständlichkeit. In Netzwerken, in denen hinter einem Interface des Routers zwei Nachbarrouter angeschlossen sind, zum Beispiel über ein „Switch“ oder „Hub“, genügt die Orientierung über Interfaces nicht mehr, da über zwei unterschiedliche Router auch zwei unterschiedliche Routen zu einem Subnetz existieren können und in diesem Fall der RIP-MTI-Algorithmus die Routen nicht

voneinander unterscheiden kann und so auch den Source-Loop nicht erkennen und das CTI-Problem nicht verhindern kann. Die Nachbarrouter können allerdings anhand ihrer IP-Adresse eindeutig identifiziert werden und somit können auch die Routen über die Router eindeutig voneinander unterschieden werden. Grundsätzlich ist es kein Problem für den RIP-MTI-Algorithmus sich an einem anderen Objekt auszurichten bzw. zu orientieren, solange es identifizierbar ist und eindeutig mit den Routen verknüpft werden kann. Der RIP-MTI-Algorithmus selbst ändert sich nicht. Bei der Implementierung richtet sich die Orientierung nach dem „Schlüssel“, mit dem die MRPM-Tabelle auf die Positionsindizes eindeutig abgebildet wird. Hier wird einfach das Interface mit der LF-IP-Adresse, also der IP-Adresse der Nachbarrouter, ausgetauscht. Da Nachbarrouter mit ihren IP-Adressen, im Gegensatz zu lokalen Interfaces, kommen und gehen können, wird hier ein weiterer Timer benötigt, der einen Eintrag in der MRPM-Tabelle löscht, wenn vom zugehörige Schlüssel, also der LF-IP-Adresse, längere Zeit keine Routing-Updates im Router mehr eingetroffen sind.

In Abbildung 5.23 ist eine Topologie mit einem Switch dargestellt, in der die drei Router R1, R3 und R4 direkt, über einen Switch, miteinander verbunden sind. Zu sehen ist hier der Source-Loop, der vom RIP-MTI-Algorithmus in Router R1 und R4 jeweils erkannt wurde. Die scheinbar alternative Route zum nicht mehr erreichbaren Subnetz 10.0.1.0/24 wird somit nicht angenommen und das CTI-Problem kann nicht entstehen.

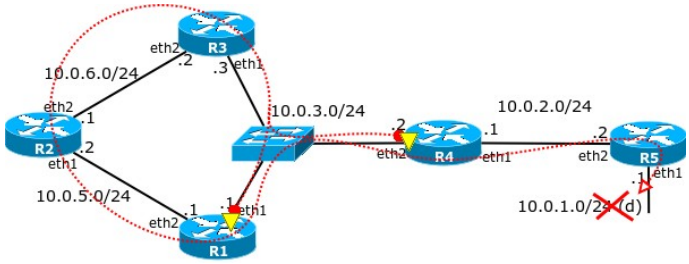


Abbildung 5.23: Topologie mit Switch

In Abbildung 5.24 ist der zur Situation zugehörige Metrikgraph abgebildet. Er unterscheidet sich nicht vom Metrikgraphen des RIP-MTI-Algorithmus mit Orientierung über Interfaces. In der Debug-Ausgabe sind die Positionsindizes der MRPM-Tabelle nun mit den IP-Adressen der Nachbarrouter verbunden. Da sich die Implementierung an den internen Positionsindizes der MRPM-Tabelle ausrichtet, ändert sich sonst nichts.

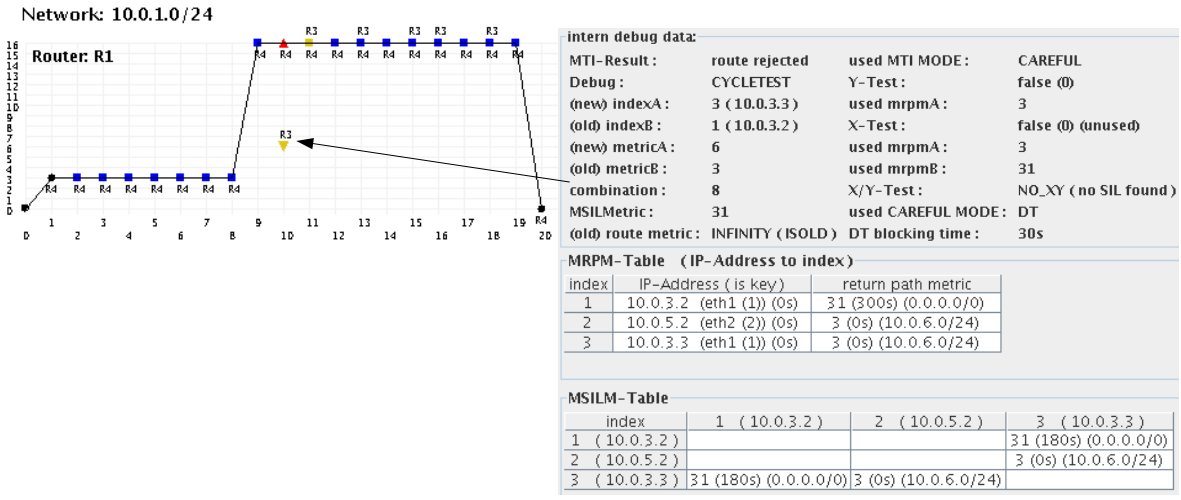


Abbildung 5.24: XTPeer Ausgabe der Orientierung über die IP-Adresse

Die Orientierung über die LF-IP-Adresse ist der Orientierung über Interfaces vorzuziehen, da letztere nicht alle möglichen Netzwerk-Szenarios abdeckt und so Lücken für das CTI-Problem lässt. Es wurde noch eine Orientierung über die Kombination von Interfaces und LF-IP-Adressen implementiert, die aber nicht benötigt wird. Sie war nur Interessant, da bei der

Orientierung über die LF-IP-Adresse festgestellt werden konnte, dass wenn am Router direkt anliegende Subnetze über eine Route von einem benachbarten Router von anderer Seite angeboten werden, als Gegenpart zur IP-Adresse des benachbarten Routers für den erkannten Simple-Loop die IP-Adresse 0.0.0.0 gewählt wurde, was letztlich nur aussagt, dass diese IP-Variable den Wert 0 besitzt. Hier könnte dann einfach die Localhost-IP-Adresse (127.0.0.1) eingesetzt werden.

5.8 Zusammenfassung und Empfehlung

In diesem Kapitel wurde die Implementierung des RIP-MTI-Algorithmus und die wichtigsten, entworfenen und implementierten Varianten vorgestellt. Einige Entwicklungen haben sich in den praktischen Überprüfungen mit der Netzwerk-Testumgebung, aus VNUML Netzwerksimulationen und XTPeer, als nicht so gut geeignet herausgestellt. Sie sind zu aufwendig in der Implementierung, beanspruchen das Routersystem zu stark oder verhindern das CTI-Problem nicht in allen Situationen. Einige können jedoch zur Weiterentwicklung und zur Übernahme in eine fertige Implementierung, die dann auch in produktiven Netzwerksystemen eingesetzt werden kann, empfohlen werden. Die hier vorgestellte Implementierung des RIP-MTI-Algorithmus wurde bislang nur innerhalb der Testumgebung aus VNUML-Netzwerksimulation und XTPeer mit wenigen, ausgefallenen Routen überprüft. Im produktiven Einsatz mit Hunderten von ausgefallenen Routen muss die Funktionalität der Implementierung des RIP-MTI-Algorithmus letztlich noch nachgewiesen werden.
Der RIP-MTI-Strict-Mode ist unter den vorgestellten RIP-MTI-Modes der Effizienteste. Er verhindert das CTI-Problem und verursacht nur eine geringe Mehrbelastung des Routersystems. Die Implementierung des RIP-MTI-Strict-Mode ist überschaubar und unkompliziert. Auf die MSILM-Tabelle könnte verzichtet werden, da er nur die MRPM-Tabelle benötigt. Allerdings ist er viel zu streng in der Überprüfung alternativer Routen und lehnt unter bestimmten Umständen auch eine große Zahl gültiger, alternativer Routen ab, wenn er keinen Simple-Loop erkennen kann. Diese Strenge kann dem RIP-MTI-Strict-Mode genommen werden. Zum einen sollte der RIP-MTI-Normal-Mode dazugeschaltet werden. Der RIP-MTI-Normal-Mode entspricht dem ursprünglich entwickelten Konzept des RIP-MTI-Algorithmus, er funktioniert allerdings nur in Netzwerken mit einer einfachen Topologie. Dafür wird die einfache MSILM-Tabelle benötigt, die jedoch nur eine geringe Mehrbelastung verursacht. Die Vollständigkeit der MSILM-Tabelle braucht hier nicht berücksichtigt zu werden. Hat ein Interface keinen Simple-Loop oder mit nur einem einzigen Interface eine Verbindung über einen Simple-Loop, steht es also nur einmal in der MSILM-Tabelle, so genügt die Vorgehensweise des RIP-MTI-Normal-Mode, der hier einen Source-Loop erkennen kann und darum gültige, alternative Routen in diesem Fall nicht ablehnen wird. Trotzdem sollte auch die Ablehnungsdauer des RIP-MTI-Normal-Mode mit einem Timer beschränkt und kontrolliert werden. Besitzt das Interface jedoch mit mehreren anderen Interfaces des Routers einen Simple-Loop, ist es also mehrfach in der MSILM-Tabelle aufgeführt, so muss die alternative Route mit dem RIP-MTI-Strict-Mode überprüft werden. Der RIP-MTI-Strict-Mode verwendet den X-/Y-Test und lehnt deswegen viele gültige, alternative Routen ab. Der X-Test sollte durch einen Y-Test mit Verzögerung der Übernahme größerer Metriken ersetzt werden, wodurch dieser etwas strenger wird und X-Kombinationen erkennen kann. Der RIP-MTI-Strict-Mode wird dann aber insgesamt weniger gültige, alternative Routen ablehnen. Als Verzögerungszeit sollten hier zwei Update-Timer Intervalle als Maximum genügen, davon ausgehend das auf allen Routern die gleichen Zeiten für den Update-Timer

eingestellt sind. Die Konvergenz des Netzwerks wird durch eine längere oder kürzere Verzögerungszeit für den RIP-MTI-Algorithmus nur unter bestimmten Umständen indirekt beeinflusst.
Um den RIP-MTI-Strict-Mode noch weiter zu verbessern sollte er um die Technik des RIP-MTI-Careful-RT-Mode erweitert werden. Wenn der RIP-MTI-Careful-RT-Mode eine alternative Route ablehnt, sendet er nach einer kurzen Zeitspanne einen RIP-Request an die Nachbarrouter, falls bis dahin kein Routing-Update mit Metrik RIP-Infinity für die Route über das Interface eingegangen ist. Die Zeitspanne sollte sich hier am Umfang der vorhanden Simple-Loops und damit am MRPM-Wert des zugehörigen Interfaces ausrichten. Auch nach der Ablehnung des in Kombination stehenden RIP-MTI-Normal-Modes sollte eine RIP-Request-Nachricht, nach Ablauf des Timers, gesendet werden.
Der RIP-MTI-Careful-DT-Mode und der -ESHC-Mode können vernachlässigt werden, da sie nicht an die Effizienz des RIP-MTI-Careful-RT-Mode heranreichen. Auch der Ansatz des LMIDT kann hier nicht zur Umsetzung empfohlen werden, da er eventuell sehr viel mehr gültige Routen ablehnen würde. Ob allerdings nicht doch ein Teil des Konzepts des LMIDT in einer endgültigen Implementierung des RIP-MTI-Algorithmus umgesetzt werden könnte, um auch vorhandene Routing-Loops anhand des CTI-Problems zu erkennen, ist noch offen. So wie der LMIDT derzeit implementiert ist bringt er keinerlei Vorteile. Eventuell wird er aber auch durch das verwenden des oldmetric_delay, bzw. dem verzögerten Y-Test, und der damit ebenfalls verbundenen verzögerten Betrachtung der Routing-Informationen, überhaupt nicht mehr benötigt.
Der RIP-MTI-Careful-RT-Mode mit einem vorgeschalteten RIP-MTI-Normal-Mode und ohne den X-Test, sondern nur mit dem verzögerten Y-Test und das „oldmetric_delay“ auf 2 Update-Intervalle eingestellt, sowie einem relativ kurzen Request-Timer, abhängig vom MRPM-Wert der zugehörigen Richtung und der Orientierung des Algorithmus über die LF-IP-Adresse wären somit eine effiziente und gleichzeitig überschaubare Erweiterung einer konventionellen RIP-Implementierung um das CTI-Problem in RIP Netzwerken zu verhindern.

6 Fazit und Ausblick

Mit dem RIP-MTI-Algorithmus steht eine abwärtskompatible Erweiterung für das Routing-Information-Protocol (RIP) zu Verfügung, die das Auftreten des CTI-Problems und die damit verbundene Bildung von Routingschleifen verhindern kann. Somit ist es möglich RIP auch in Netzwerken mit komplexeren Topologien einzusetzen, bzw. ein vorhandenes Netzwerk, in dem RIP bereits eingesetzt wird, gefahrlos um neue Verbindungen zu erweitern. Die Theorie des RIP-MTI-Algorithmus, mit Hilfe der Metrik der Routen Netzwerkschleifen und Routingschleifen erkennen und voneinander unterscheiden zu können und so das CTI-Problem zu verhindern, konnte in dieser Arbeit bestätigt werden. Allerdings konnte auch festgestellt werden, dass die Umsetzung dieser Theorie in eine funktionierende Implementierung, die das CTI-Problem zwar immer verhindert, aber gleichzeitig nicht schlechter ist als die Implementierungen des konventionellen RIP-Algorithmus, nicht einfach ist. Hier gibt es viele Möglichkeiten, das gleiche Verhalten über unterschiedliche Vorgehensweisen zu erreichen. Verschiedene Varianten des RIP-MTI-Algorithmus wurden hier implementiert, die auch Schwächen im Konzept aufgezeigt haben. Letztlich konnte jedoch mit dem RIP-MTI-Careful-RT-Mode eine Variante der Implementierung des RIP-MTI-Algorithmus empfohlen werden, die einerseits das CTI-Problem verhindert und bei entsprechender Implementierung andererseits nicht schlechter als der konventionelle RIP-Algorithmus arbeitet.

Aber auch wenn der RIP-MTI-Algorithmus als Erweiterung zu RIP letztlich akzeptiert werden sollte und sich die Verhinderung des CTI-Problems auch weiterhin bestätigen lässt, klafft eine große Lücke zwischen RIP und den moderneren Routing-Protokollen. Moderne Routing-Protokolle sind RIP nicht nur bei der Verhinderung von Routingschleifen überlegen. Hierfür verwenden viele eindeutige Identifikationsnummern, die Router und Routen erhalten, um Routing-Informationen zu identifizieren und ältere Informationen von aktuelleren unterscheiden zu können. Zusätzlich wird die Weitergabe von Routing-Informationen und die Bestätigung der Verbindungen zwischen den Routern getrennt behandelt. Für letzteres werden zum Beispiel eigene Hello-Nachrichten verwendet, die keine Routing-Informationen enthalten. Diese Routing-Protokolle bieten eine formal bewiesene Schleifenfreiheit und eine schnellere Konvergenz des Netzwerks nach dem Ausfall einer Route.

Der große Vorteil von RIP ist dessen Einfachheit. RIP-Router benötigen beim Start keine Parameter, wie zum Beispiel eine eindeutige Router-ID, die entsprechend verwaltet und im Routing-Prozess mit einbezogen werden muss. Der RIP-Algorithmus benötigt im Vergleich zu anderen Routing-Algorithmen nur relativ wenige Zeilen Quellcode, was übersichtliche Implementierungen ermöglicht, die somit auch sehr schnell und einfach robust zu bekommen sind. RIP ist daher die erste Wahl für kleine, einfache oder auch experimentelle Netzwerke, in denen ein einfacherer, schnell verfügbarer oder in seinem Tun leicht nachvollziehbarer Routing-Algorithmus benötigt wird.

Mit Hilfe des RIP-MTI-Algorithmus könnte die maximale Distanz von 15 Hops und 16 als RIP-Infinity aufgehoben werden, da das CTI-Problem und Routingschleifen nicht mehr zu befürchten sind. Allerdings verliert eine solche Erweiterung dann auch bereits schon die Abwärtskompatibilität zur RIP-Standard-Spezifikation.

Generell bietet RIP allerdings noch genug Potential, um das Protokoll weiter auszubauen und an moderne Gegebenheiten anzupassen, allerdings auf Kosten der Kompatibilität zur bestehenden Standard RIP-Spezifikation.

Literaturverzeichnis

[Sch99] Andreas Schmid, RIP-MTI: Minimum-effort loop-free distance vector routing algorithm, Diplomarbeit, 1999, Universität Koblenz-Landau

[STD53] G. Marlin, RIPv2, 1998, IEEE RFC

[Kle01] Thomas Kleemann, RIPeval - Evaluierung und Weiterentwicklung des RIP-MTI-Algorithmus, Diplomarbeit, 2001, Universität Koblenz

[Koc05] Tobias Koch, Implementation und Simulation von RIP-MTI, Diplomarbeit, 2005, Universität Koblenz

[Päh06] Daniel Phäler, Extern steuerbare Routing-Updates im RIP-Daemon der Quagga-Programmsuite, Diplomarbeit, 2006, Universität Koblenz

[Lan07] Stefan Lange, Zentrale Betrachtung von Routing-Informationen zur Analyse des Konvergenzverhaltens verschiedener RIP-Algorithmen und Unterstützung des Generierens von Testfällen, Diplomarbeit, 2007, Universität Koblenz

[Keu07] Tim Keupen, Generierung von Testfällen für den RIP-MTI Algorithmus, Diplomarbeit, 2007, Universität Koblenz

[Wol06] Bernhard Wolf, Untersuchung und Simulation des RIP-MTI-Algorithmus, Studienarbeit, 2006, Universität Koblenz

[Bel57] R. E. Bellman, Dynamic Programming, 1957, Princeton University Press

[FoFu62] L.R. Ford, D.R. Fulkerson, Flows in Networks, 1962, Princeton University Press

[FloJa94] Sally Floyd, Van Jacobsen, The Synchronization of Periodic Routing Messages, 1994, Lawrence Berkeley Laboratory

[RFC1058] C. Hedrick, RFC 1058 - Routing Information Protocol, 1988, IEEE RFC

[RFC1812] F. Baker, RFC 1812 - Requirements for IP Version 4 Routers, 1995, IEEE RFC

[Dik06] Jeff Dike, User Mode Linux, 2006, Pearson Education Inc.

[PeDa04] Larry L. Peterson, Bruce S. Davie, Computernetze, 2004, Dpunkt Verlag, 3rd Edition (dt)

[Fra05] Leif Franker, Vergleichende Simulation von RIP und RIP-MTI, Studienarbeit, 2005, Universität Koblenz

[Tan03] Andrew S. Tanenbaum, Computernetzwerke, 2003, Prentice Hall, 3rd Edition (dt)

[Kan00] Olaf Kandel, RIP-MTI - Simulation des erweiterten RIP-Algorithmus, Diplomarbeit, 2000, Universität Koblenz

[Hui99] Christian Huitema, Routing in the Internet, 1999, Prentice-Hall, 2nd Edition

[StDiKe08] Ch. Steigner, H. Dickel, T. Keupen, RIP-MTI: A New Way to Cope with Routing Loops, 2008, Seventh International Conference on Networking (icn 2008)

[StSch01] Ch. Steigner, Andreas Schmid, Avoiding Counting to Infinity in Distance Vector Routing, 2001, Kluwer Academic Publishers

Anhang A: Funktionen des RIP-MTI-Daemons

Im Folgenden sind die wesentlichen Funktionen der Implementierung des RIP-MTI-Daemons kurz erläutert.

```
void rip_mti_init(void)
```
Initialisierung der globalen Variablen des RIP-MTI-Algorithmus.

```
void mti_sil_timeout(void);
```
Hier wird überprüft, ob einzelne Simple-Loops in der MSILM-/MRPM-Tabelle längere Zeit nicht mehr bestätigt werden konnten und deshalb nicht mehr gültig sind und gelöscht werden müssen, bzw. die Metrik auf RIP-MTI-Infinity gesetzt werden muss. Diese Funktion wird derzeit bei jedem Betreten des RIP-MTI-Algorithmus aufgerufen.

```
int rip_mti_cycleok(unsigned int indexA, unsigned int indexB,
 struct rip_info *, struct rte *, struct interface *,struct sockaddr_in *);
```
Hier ist die Kernfunktionalität der Simple-Loop Erkennung und Source-Loop Vermeidung implementiert. Alle RIP-MTI-Modes des RIP-MTI-Algorithmus (Normal, Strict, usw ...) sind in dieser Funktion zu finden.

```
int rip_mti_routeok(struct rip_info *, struct rte *, struct interface*,
 struct sockaddr_in *,u_int32_t);
```
Diese Funktion stellt die Verbindung zwischen der Implementierung des konventionellen Quagga RIP-Daemons und der Implementierung des RIP-MTI-Algorithmus dar. Sie wird innerhalb des Routing-Entscheidungsprozesses aus der Datei ripd.c aufgerufen.

```
void rip_mti_set_table_entries(u_int32_t,u_int32_t,u_int32_t,
 struct in_addr,int );
```
Hier werden neu gefundene Simple-Loops in die MRPM und MSILM-Tabelle eingetragen.

```
struct rip_mti_entry * rip_mti_msilm_table_lookup(u_int32_t, u_int32_t);
```
Diese Funktion fragt nach einem Simple-Loop zwischen zwei Interfaces, bzw. zwei Einträgen der MRPM-Tabelle. Übergeben werden die Indizes der Einträge der MRPM-Tabelle.

```
struct rip_mti_entry * rip_mti_mrpm_lookup (u_int32_t );
```
Gibt den MRPM-Wert eines MRPM-Eintrags anhand des Index zurück.

```
int xy_combination_test(int,unsigned int, unsigned int, u_int32_t,
 u_int32_t, u_int32_t);
```
Dies ist der X/Y-Test des RIP-MTI-Algorithmus.

```
void xtpeer_sl_readSimpleLoopTables(void);
```
Über diese Funktion wird die Information aus der MSILM- und MRPM-Tabelle ausgelesen und in den Sende-Buffer des SL-Clients im RIP-MTI-Daemon geschrieben, um sie an den SL-Server des XTPeers zu senden.

```
void xtpeer_sl_finish_MTI_Debug_Info_Message(int)
```
Diese Funktion schließt das Schreiben der Informationen des RIP-MTI-Algorithmus in den Sende-Buffer des SL-Clients ab. Im Anschluss wird eine MTI_INFO_DEBUG_MESSAGE mit den protokollierten Status-Informationen an den XTPeer gesendet.

```
void autoconfigure(int );
```
Diese Funktion implementiert die globale Selbst-Konfiguration des RIP-MTI-Algorithmus. Sie wird über die Funktion rip_mti_sil_timeout() aufgerufen und wählt anhand der übergebenen Anzahl an vorhandenen Simple-Loops den entsprechenden RIP-MTI-Mode.

```
int ip2index(struct in_addr, unsigned int);
```
Abbildung der Learned-From-IP-Adresse (LF-IP-Adresse) auf einen Index der MRPM-Tabelle. Ermöglicht die Orientierung des RIP-MTI-Algorithmus über LF-IP-Adressen.

```
int if2index(struct in_addr, unsigned int);
```
Abbildung des Interface auf einen Index der MRPM-Tabelle. Ermöglicht die Orientierung des RIP-MTI-Algorithmus über Interfaces.

```
int ipif2index(struct in_addr, unsigned int);
```
Abbildung der Kombination aus Interface und LF-IP-Adresse auf einen Index der MRPM-Tabelle. Ermöglicht die Orientierung des RIP-MTI-Algorithmus über beide Möglichkeiten der Identifizierung einer Route.

```
void resetMTIModesAndTimers(struct rip_info *);
```
Diese Funktion setzt durch den RIP-MTI-Careful-Mode gesetzte Flags der Markierung ausgefallener Routen zurück und stoppt laufende Threads der Careful-Modes.

```
void resetMTI(void);
```
Diese Funktion löscht den Inhalt sämtlicher Daten-Strukturen der Implementierung des RIP-MTI-Algorithmus und versetzt ihn so wieder in den initialen Zustand, um zum Beispiel eine Simulation von vorne zu beginnen. Sie wird über die nachfolgende Funktion aufgerufen, die in der Datei ripd.c implementiert ist.

```
int xtpeer_cleanRoutingTable(void);
```
Diese Funktion wird vom XTPeer aus aufgerufen und löscht den Inhalt der Routing-Tabelle Gleichzeitig wird auch die obige Funktion zur Löschung der Daten-Strukturen des RIP-MTI-Algorithmus aufgerufen. Das Ausführen der Funktion soll einen umständlichen Neustart der einzelnen RIP-Daemons der Simulation überflüssig machen.

```
int rip_mti_lmidt_routeok(unsigned int,struct rip_info *, struct rte *,
 struct sockaddr_in *);
```
Diese Funktion ruft den LMIDT auf. Hier wird überprüft, ob die Metrik einer Route um die Metrik eines zugehörigen Simple-Loops gewachsen ist.

```
void rip_mti_lmidt_cti_break(u_int32_t,struct rip_info *, struct rte *,
 struct sockaddr_in *);
```
Diese Funktion wird vom LMIDT aufgerufen, um die neue Route zurückzuhalten und die gegenwärtige Route in der Routing-Tabelle mit der Metrik RIP-Infinity zu versehen.

```
unsigned int getMaxLoopMetric(unsigned int);
```
Über diese Funktion wird der größte MSILM-Wert eines Interfaces, bzw. einer LF-IP-Adresse gefunden. Übergeben wird der Index des Eintrags der MRPM-Tabelle. Der Rückgabewert ist der höchste MSILM-Wert der mit dem Index in Verbindung steht.

```
void sendIntoLoopUnicast(unsigned int);
```
Sendet über alle zum Index gehörenden Simple-Loops ein Routing-Update, via Unicast an die IP-Adresse des Nachbarrouter. Übergeben wird der Index des Eintrags der MRPM-Tabelle.

```
void sendIntoLoopMulticast(unsigned int index);
```
Sendet über alle zum Index gehörenden Simple-Loops ein Routing-Update, via Unicast an die IP-Adresse des Nachbarrouter.

```
unsigned int getTimerTimeoutTime(int, int, int);
```
Diese Funktion berechnet den Timer für die verschiedenen Konfigurationsmöglichkeiten. Übergeben wird hier der Integerwert des Protokollworts (update, maxmsilm, mrpm, value), der Zeitwert, sowie der Index des Eintrags der MRPM-Tabelle.

```
struct dt_route_data * buildDenyTimerRouteData(u_int32_t ,struct rte *,
 struct sockaddr_in *, struct rip_info *);
```
Hier werden für den RIP-MTI-Careful-DT-Mode die Daten einer abgelehnten, alternativen Route zwischengespeichert. Das C-Konstrukt „dt_route_data“ wird einem Thread von bestimmter Dauer übergeben, der die Route nach seinem Ablauf erneut an den Routing-Prozess übergibt, aber zuvor die ausgefallene Route in der Routing-Tabelle markiert.

```
struct request_data * buildRequestData(struct index_entry * ,struct rte *,
 struct rip_info *);
```
Diese Funktion speichert für den RIP-MTI-Careful-RT-Mode einige Daten der abgelehnten, alternativen Route ab, um RIP-Request-Nachrichten gezielt, nach dem nicht mehr zu erreichenden Subnetz, zu stellen.

```
struct sockaddr_in * createSockAddr_In(struct in_addr);
```
Dies ist eine Hilfsfunktion um das sockaddr_in-C-Konstrukt zu erstellen. Es wird für das Senden von Nachrichten via Unicast benötigt.

```
void requestESHCMulticast(struct rip_info *);
```
Diese Funktion gehört zum RIP-MTI-Careful-ESHC-Mode und ermöglicht das Senden von Requests via Multicast über die Interfaces, die in der ESHC-Liste indirekt aufgerührt sind.

```
void requestESHCUnicast(struct rip_info *);
```
Diese Funktion gehört zum RIP-MTI-Careful-ESHC-Mode und ermöglicht das Senden von Requests via Unicast an die Nachbarrouter, die in der ESHC-Liste indirekt aufgerührt sind.

Anhang B: Erweiterung der XT-Protokoll-Syntax

Im Folgenden ist die Erweiterung der XT-Protokoll-Syntax in EBNF beschrieben, um den Status des RIP-MTI-Algorithmus über den XTPeer beeinflussen zu können. Über die XT-Client/Server Struktur werden die Kommandos vom XTPeer an den Quagga RIP-MTI-Daemon gesendet. Der RIP-MTI Daemon bestätigt den Empfang und die Übernahme der Konfiguration stets mit einem „*DONE*“ oder mit den abgefragten Informationen, wenn zum Beispiel „*SHOW_STATUS“* verwendet wird.

XT-Client_Nachricht := <MTI> | andere XTPeer-Befehle

MTI := "*MTI*" (<MODE> | <ORIENTATION> | <ACTIVITY> | <LMIDT> | <RUNTIMER> | <AUTOCONFIG> | <MSILM> | <TRIGGER_INTERVAL> | <OLDMETRIC_DELAY> | <VERSION> | <SHOW_STATUS>)

MODE := "*MODE*" ("*OFF*" | "*LISTEN*" | "*NORMAL*" | "*STRICT*" | <CAREFUL> | "*SHOW_STATUS*")

CAREFUL := "*CAREFUL*" (<ESHC> | <DT> | <RT>)

ESHC := "*ESHC*" ("*OFF*" | "*PASSIVE*" | "*ACTIVE*")

DT := "*DT*" <TIMER>

RT := "*RT*" <TIMER>

TIMER := ("*OFF*" | "*VALUE*" int | "*MAXMSILM*" int | "*MRPM*" int | "*UPDATE*" int | "*SHOW_STATUS*")

int := $n \in \mathbb{N}$, $1 \leq n$

ORIENTATION := "*ORIENTATION*" ("*IP*" | "*INTERFACE*" | "*IP_IF*" | "*SHOW_STATUS*")

ACTIVITY := "*ACTIVITY*" ("*PASSIVE*" | "*ACTIVE*" | "*SHOW_STATUS*")

LMIDT :="*LMIDT*" (("*MSILM*" | "*MRPM*") <TIMER> | "*SHOW_STATUS*")

RUNTIMER := "*RUNTIMER*" (<TIMER> | "*SHOW_STATUS*")

AUTOCONFIG := "*AUTOCONFIG*" ("*OFF*" | "*GLOBAL*" | "*ROUTE*" | "*SHOW_STATUS*")

OLDMETRIC_DELAY := "*OLDMETRIC_DELAY*" <TIMER> | "*SHOW_STATUS*")

MSILM := "*MSILM*" ("*RECV*" | "*CALC*" | "*SHOW_STATUS*")

TRIGGER_INTERVAL := "*TRIGGER_INTERVAL*" ("*VALUE*" int | "*SHOW_STATUS*")

VERSION := "*VERSION*"

Anhang C: Erweiterung der SL-Protokoll-Syntax

Im Folgenden ist die Erweiterung der SL-Protokoll-Syntax in EBNF beschrieben, um die internen Informationen des RIP-MTI-Algorithmus mit Hilfe des XTPeers protokollieren zu können. Über die SL-Client/Server Struktur werden die Log-Meldungen vom Quagga RIP-MTI-Daemon an den XTPeer gesendet, wo sie ausgewertet, angezeigt und auch gespeichert werden können. Die MTI_DEBUG__INFO_MESSAGE wurde zwar bereits in [Lan07] eingeführt, aber in dieser Ausarbeitung erheblich erweitert.

MTI_DEBUG_MESSAGE := "*MSG_MTI_DEBUG_INFO*" !-! uptime !-! subnet_prefix !-! subneth_mask_length !-! (old)interface(name) !-! (old)interface(index) !-! (old)learnedfromIP !-! (old)route_metric !-! (old)old_interface(name) !-! (old)old_interface(index) !-! (old)old_learnedfromIP !-! (old)_oldroute_metric !-! (new)interface(name) !-! (new)interface(index) !-! (new)learnedfromIP !-! (new)route_metric !-! <ORIENTATION> !-! indexA !-! indexB !-! <INTERN_DEBUG_DATA> !-! mti_result !-! <MRPM-TABLE> !-! <MSILM-TABLE>

ORIENTATION := ("*IP*" | "*IF*" | "*IP_IF*")

INTERN_DEBUG_DATA := (<RUNTIMER_ACK> | <LMIDT >| <LMIDT_ACK> | <EQUAL_INDICES> | <CAREFUL_ACK> | <HIGH_METRIC> | <CYCLETEST>)

RUNTIMER_ACK := "*RUNTIMER_ACK*" !;! runtime

LMIDT := "LMIDT" [!;! vector_position !;! msilm !;! timeout_timer]

LMIDT_ACK := "*LMIDT_ACK*" (!;! "*FAIL*" | ("*METRIC_UPDATE*" !;! (new)route_metric !;! (new)learnedfromIP) | "*PASS*")

EQUAL_INDICES := "*EQUAL_INDICES*"

CAREFUL_ACK := "*CAREFUL_ACK*" ([!;! "*DT*" (!;! "*FAIL*" | ("*METRIC_UPDATE*" !;! (new)route_metric !;! (new)learnedfromIP | "*PASS*")]) | ([!;! "*ESHC*" (!;! (index { !;! index } !;! "*FAIL*") | "*PASS*")])

HIGH_METRIC := "*HIGH_METRIC*" !;! (new)route_metric !;! (old)route_metric

CYCLETEST := "*CYCLETEST*" !;! indexA !;! indexB !;! metricA !;! metricB !;! combination !;! msilm (!;! (<ISOLD> | <NOT_ISOLD>))

ISOLD := "*ISOLD*" !;! (<LISTEN> | <NORMAL> | <STRICT> | <CAREFUL>)

NOT_ISOLD := "*NOT_ISOLD*" (!;! NOT_MIN | !;! XY_CHECK [!;! SIL_UPDATE])

LISTEN := "*LISTEN*" !;! <XY_CHECK>

NORMAL := "*NORMAL*"

STRICT := "*STRICT*"!;! <XY_CHECK>

CAREFUL := "*CAREFUL*" !;! <XY_CHECK> [!;! "*DT*" [!;! blocking_timer]] | [!;! "*ESHC*" { !;! index }] | [!;! "*RT*" [!;! blocking_timer]]

XY_CHECK := (("*Y*" !;! y !;! (mrpmB | mrpmA)) | "*EQUAL_METRICS*") !;! ("*X*" !;! x !;! used !;! mrpmA !;! mrpmB) [!;! "*NO_XY*"]

SIL_UPDATE := "*SIL_UPDATE*" !;! mrpmA !;! mrpmB !;! msilm !;! subnet_prefix !;! subnet_mask_length

MRPM-TABLE := "*MRPM*" { !;! index !;! mrpm !;! timer !;! subnet_prefix !;! subnet_mask_length }

MSILM-TABLE := "*MSILM*" { !;! vector_position !;! msilm !;! timer !;!subnet_prefix !;! subnetmask_length}