Gullivers Reisen durch FF-Routistan - oder: Vom Fußgängerzonenproblem zur IGP-Kopplung

Contents

(Lesen auf eigene Gefahr! Aber viel Spaß dabei!-)

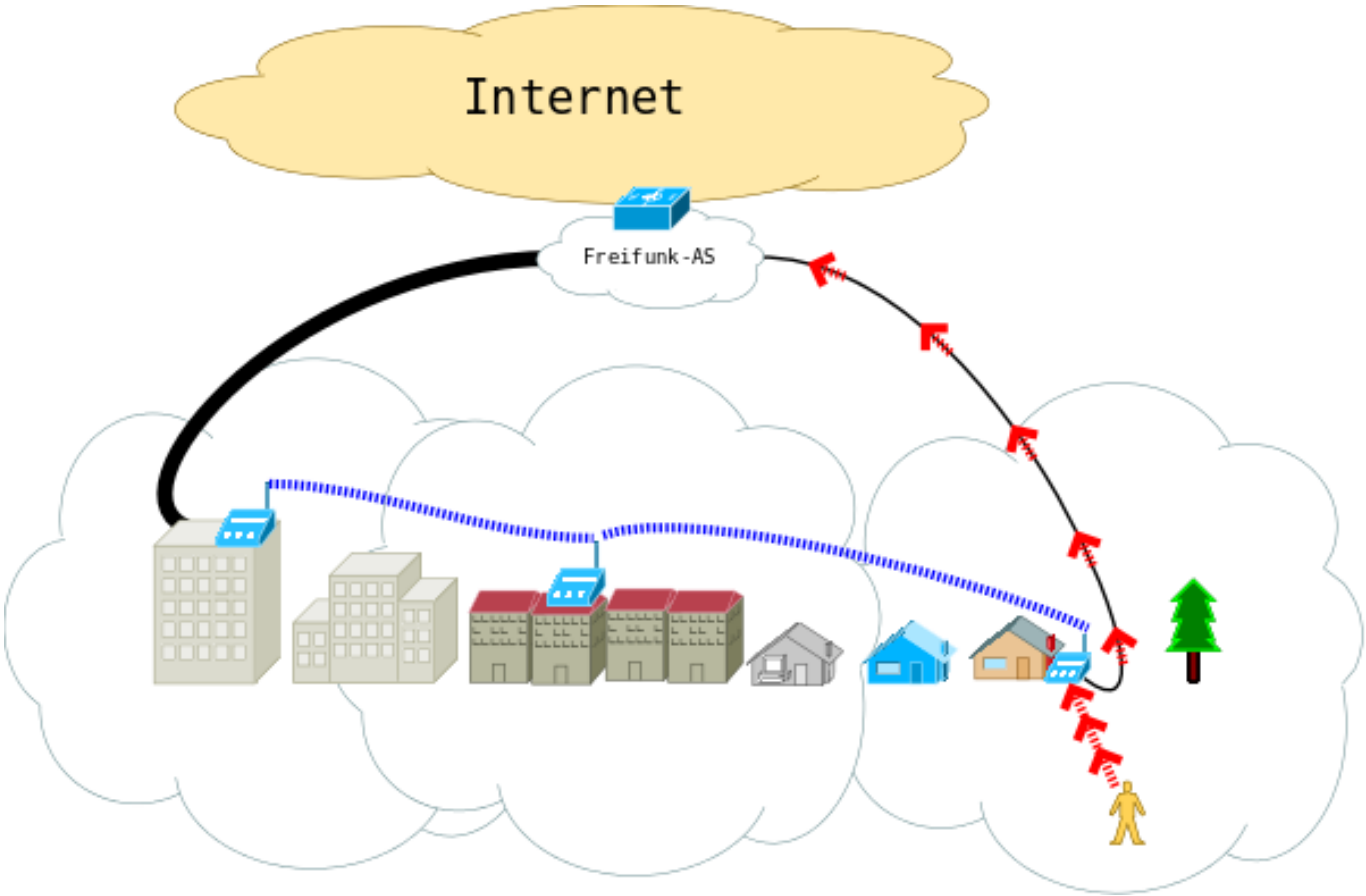
Es gibt auch eine Zusammenfassung dieses Textes.

Freifunk - das erste Problem: WLAN mesh Chaos

Das offensichtlichste Routingproblem des Freifunk besteht darin, unter den schwierigen Randbedingungen eines WLAN adhoc Netzes optimales Routing zu erreichen. Im Gegensatz zu einer Kabelverbindung ändern sich in einem WLAN-Netz die vorhandenen Verbindungen und die Qualität der Verbindungen häufig. Dieses Problem ist mittlerweile für die Praxis ausreichend gut gelöst. Ein wesentlicher Schritt dazu waren die Entwicklung von angepassten Routingprotokollen (OLSR, B.A.T.M.A.N., Babel, . . .) sowie von Metriken (insb. ETX).

Freifunk - das zweite Problem: die lange Fußgängerzone

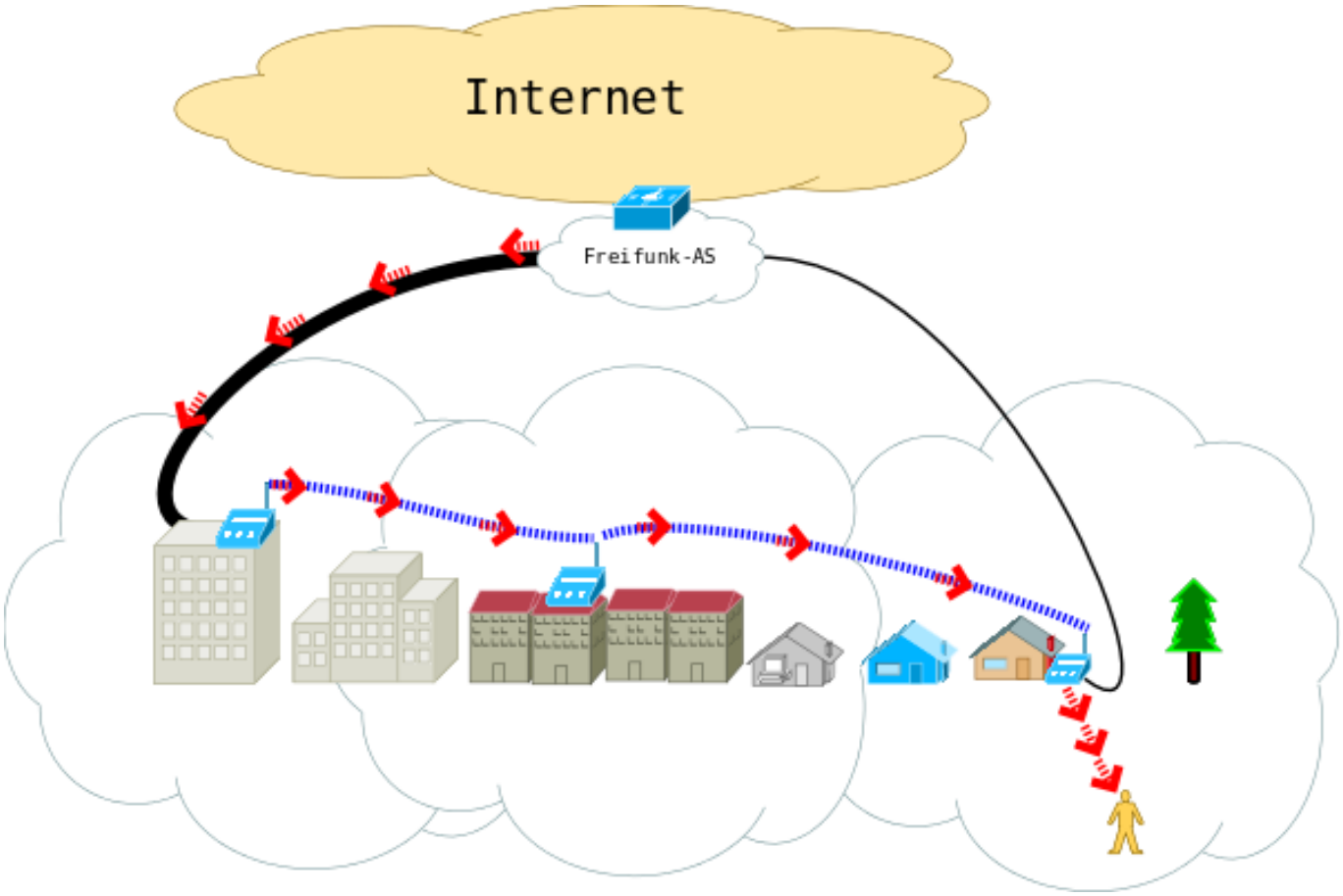
Weniger offensichtlich ist das Problem der entkoppelten Metriken, eher bekannt unter dem Namen “Fußgängerzonenproblem”. Betrachten wir als Beispiel eine längere Fußgängerzone, die nur an beiden Enden Uplinks in ein FF-Backbone Netz hat.



Zwischen diesen beiden Enden zieht sich ein WLAN mesh die Straße entlang. Befindet sich nun ein WLAN client am unteren Ende der Straße, so werden die von ihm ausgehenden Datenpakete Richtung Internet von dem FF-Router, an

dem der client hängt, über den kürzesten Weg gemäß der mesh-Metrik Richtung Uplink geroutet. D.h. sie gehen über den Uplink am unteren Ende der Straße raus, was in dieser Situation auch optimal ist.

Bleibt aber die Frage nach dem retour traffic, der aus dem Internet zum client geht? Diese Frage ist übrigens nicht nur von akademischem Interesse, sondern auch von praktischem, da der download üblicherweise das Gros des client traffics ausmacht (Konsumiere!). Mit der Verbreitung von Freifunk entstehen auch größere Meshes, in denen suboptimales Routing bis zur Ungenießbarkeit führen kann. Und zwar dann, wenn das Routing im Backbone nicht die Metrik des Meshes berücksichtigt.



Der umgekehrte Fall ist quantitativ vernachlässigbar, wg. kaum upstream traffic. Außerdem hat ein Backbone deutlich mehr Bandbreite als die Uplinkverbindungen (VPN), die das Mesh mit dem Backbone verbinden. Hat zB der Uplinkrouter am oberen Ende der Straße einen dickeren Uplink (50/10 Mbit) und der am unteren Ende einen kleineren (10/1 Mbit), so wird traffic aus dem Backbone des Freifunk-AS vor allem über das obere Ende der Straße ins Mesh kommen - und dann nach längerer, verlustreicher Reise den client am unteren Ende der Straße erreichen. (Im Bild ist wegen der Übersichtlichkeit nur ein Mesh-Router auf der Strecke dargestellt. Das könnten aber auch 10, 20 oder noch mehr „Zwischenstationen“ sein.)

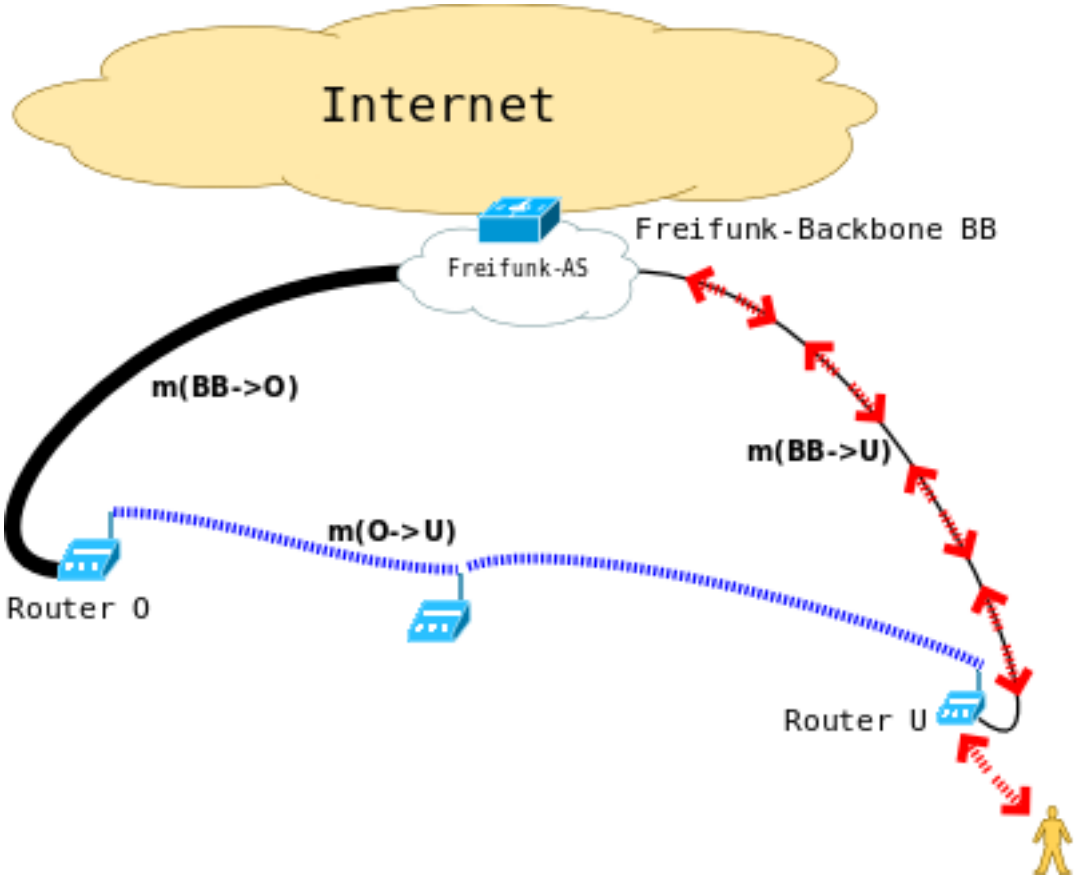
Diese Situation setzt voraus, dass der Backbone die jeweiligen Metriken zwischen Backboneroutern und Uplinkroutern berücksichtigt, zB indem die Metrik die Bandbreite des jeweiligen Uplinks oder die Latenz einbezieht. Das ist aber ohnehin sinnvoll, um zumindest die maximale einzelne downstream Bandbreite des Meshes auszunutzen.

Wenn die Metrik des Backbones keine Uplinks mit einbezieht, geschieht das Routing in Richtung Mesh eben zufällig, d.h. mal hat dieser, mal jener downstream traffic zum client das “Fußgängerzonenproblem”.

Eine gute Lösung für dieses Problem muss also dafür sorgen, dass die Metrik des Meshes auch im Routing des Backbones berücksichtigt wird. Aber kein vernünftiger (d.h. kommerzieller) Backbonebetreiber wird sich das Gezappel, das in einem WLAN-Mesh an Routinginformationen unterwegs ist, in den Backbone holen wollen. Also keine Hoffnung :-(... außer auf eigene FF-Backbones!

Fußgängerzonenproblem - erste Lösung: 1 IGP für das ganze FF-Netz

Die offensichtlichste Lösung für das Fußgängerzonenproblem besteht darin, im kabelbasierten Netz (FF-Backbone) das gleiche Routingprotokoll zu fahren wie in den WLAN-Meshes. Diese Methode wird zZ in den Communities des FF-Rheinland verwendet. Dass zZ im FF-RL das Routing auf Layer 2 abläuft (B.A.T.M.A.N. advanced) und nicht auf Layer 3 (wie in Berlin mit OLSR), ändert nichts am Grundproblem.



Durch die Ausdehnung des Routings in einem WLAN-Mesh auf das gesamte (FF-)Netz wird das Fußgängerzonenproblem so gelöst:

- der Router mit dem WLAN-client am unteren Ende der Straße schickt die von seinem client ausgehenden Pakete (bzw. frames bei L2-Routing) entsprechend der Metrik der (default-)Route, also wie bisher normalerweise über seinen Kabeluplink, da er von dort die (default-)Route mit der besten Metrik bekommt.
- Umgekehrt schicken die Router des FF-Backbone Pakete für den client zum Router am unteren Ende der Straße, da dieser die Route zum client mit der besten Metrik in Richtung FF-Backbone announced.

Im Beispiel hat zwar der untere Router (U) einen schwächeren Uplink zum Backbone (BB) als der Router (O) am oberen Ende (m(BB->U) > m(BB->O)), aber dafür kommt die Metrik für die Durchquerung der Fußgängerzone per WLAN hinzu, sodass (bei geeigneter Kostenfunktion m: Linkqualität -> Metrik) insgesamt gilt: m(BB->O) + m(O->U) > m(BB->U). Das ist aber keine Magie, sondern Designziel eines IGP, dass es innerhalb seiner Routingdomäne optimales Routing herstellt.

Je besser ein IGP und seine Metrik mit der Heterogenität der Übertragungsmedien (WLAN vs. Kabel) klarkommt, um so optimaler ist auch das globale Routing in diesem FF-Netz (= Backbone + Uplinks + Meshes). (ZB hat Babel differenziertere Möglichkeiten für die Metrik als B.A.T.M.A.N.) Weil optimales Routing in einem WLAN adhoc Mesh ein schwierigeres Problem ist als das Routing im Kabel, nimmt man sinnvollerweise das in den meshes verwendete IGP auch für's Kabel (FF-RL: B.A.T.M.A.N.). Umgekehrt liefert zB OSPF zwar ein prima Routing in großen Kabelnetzen, aber in einem WLAN adhoc mesh taugt es in der Praxis nicht, da es bestenfalls (im Point-to-Multipoint Modus) eine hop count Metrik verwenden kann. Und die kann geradezu fatal sein, weil "dank" WLAN dabei auch der link mit den übelsten Paketverlusten vom OSPF gern genommen wird, solange die Routinginformation noch gelegentlich mit einem hop durch die Luft kommt.

Diese erste, einfache Lösung hat aber auch ihre Schattenseiten:

- Bei L2-Routing zB die riesige Broadcastdomäne, die B.A.T.M.A.N. über das ganze betrachtete Netz herstellt.
- Außerdem benötigt jeder Router auch für jeden client die Information, mit welchem Router dieser verbunden ist. Wie in der Realität zu beobachten, führt dies neben überwiegend unnützem Braodcasttraffic auch zu mehr Routinginformation. Und diese wird bei DV-Routing regelmäßig in eher kurzen Abständen durchs Netz geschickt, und trägt sogar mehr zum Grundrauschen bei, als der Broadcasttraffic der clients.

Broadcastrauschen kann man durch die Verwendung von L3-Routing vermeiden, und den Traffic des Routingprotokolls insb. durch Verwendung eines link-state Protokolls minimieren. Beides zusammen erreicht man zB mit OLSR.

Link-state hat in diesem Szenario aber auch eine Schattenseite:

- der Berechnungsaufwand für eine Topologie wächst bei guter Implementierung mit r * log(r) + v, wobei r die Anzahl der Router und v die Anzahl aller direkten Verbindungen zwischen Routern bezeichnet (s. Wikipedia). Konkret kann das bedeuten, dass ein Router in einem 10 mal größeren Netz das 30fache an Berechnungszeit braucht, um den optimalen SPF-Spannbaum zu allen anderen Routern zu ermitteln.

Und je größer das Netz und vor allem der Anteil an WLAN adhoc meshes darin ist, desto häufiger ändert sich die Topologie. (Zumindest im vorigen Jahrzehnt führte dieser Zusammenhang zur (auch thermischen;) Überlastung der FF-Plasterouter in großen OLSR Netzen.)

Zusammenfassung: 1 IGP für ein ganzes FF-Netz funktioniert, aber skaliert nicht beliebig.

Fußgängerzonenproblem - zweite Lösung: 2 IGPs

Offenbar kaum beachtet ist in der Praxis bisher ein alternativer Denkansatz geblieben, nämlich die Trennung eines FF-Netzes in zwei Routingdomänen:

- ein IGP wird im Kabelteil des Netzes gesprochen,
- das andere IGP in dem oder den WLAN meshes.

Der Grund für diese Unterbelichtung dürfte vor allem darin liegen, dass es bisher keine einfache Möglichkeit gibt, Routinginformationen zwischen FF-relevanten IGPs auszutauschen. Frühere Bemühungen waren das OLSRv1-plugin für Quagga sowie OLSRv1 als Teil von XORP. Letztere verwendet nur die einfache Metrik des "offiziellen" OLSRv1 (RFC 3626) und ist daher in WLAN adhoc Meshes praktisch unbrauchbar - denn ohne ETX Metrik o.dgl. geht in dem adhoc Chaos des WLAN Gewabers leider nix. Echte Hoffnung verspricht aber die zZ (Anfang 2016) laufende Integration von Babel in Bird.

IGPs, die für WLAN adhoc Meshes entworfen werden, hatten traditionell eher schwache Unterstützung für externe Routen. Verständlich, da viel mehr als die default Route nicht gebraucht wurde. Außerdem hatte wohl niemand den Plan, Transittraffic über ein WLAN Gewaber zu schicken.

Beispiel: die externen Routinginformationen bei OLSRv1 ("HNA") enthalten nur das Ziel der Route, aber nicht die Entfernung dorthin (Metrik). Ein OLSRv1 Mesh macht daher, wenn es mit einem anderen IGP "peert", immer "hot potato" Routing (auch "best exit" genannt). Das kann aber in straßenüblichen FF-meshes dazu führen, dass der ausgehende Traffic eines clients zu einem Uplink router geht, der bereits einen ziemlich verstopften Uplink hat. Wenn dieser verstopfte Uplink so grade noch die gelegentlichen Routinginformationen durchkriegt, aber zB die zu einem Stream gehörigen ACK Pakete eines clients im oberen Sekundenbereich verzögert oder gar verwirft, dann entsteht Frust beim client. Insb. dann, wenn er wüsste, dass er über einen Nachbarrouter problemlos konsumieren könnte, dieser aber leider eine etwas größere (OLSR-ETX-)Metrik hat als der Router mit dem überlasteten Uplink - aber dafür einen dicken freien Uplink.

Dieses "hot potato Problem" kann glücklicherweise mit OLSRv2 (RFC 7181) vermieden werden, denn dort kann die Metrik externer Routinginformationen importiert werden. Bei erfolgreicher Integration von Babel in Bird (s.o.) wäre der Import externer Metriken ebenfalls möglich. (Eine sogar für OLSRv1 gangbare Variante wäre es, die Grenze zwischen den IGPs nicht in den Uplinkroutern an der Straße zu ziehen, sondern in den VPN-Endpunkten, zB "Supernodes". Aber dann hat man den für ein WLAN adhoc Mesh unvermeidlichen Routingtraffic auf den Uplinks.)

Zusammenfassung: Die Möglichkeit, zwei oder mehr IGPs zu verwenden, um das Fussgängerzonenproblem zu lösen, hängt von der Fähigkeit der beteiligten IGPs ab, externe Routinginformationen zu nutzen.

(BTW, auch bei B.A.T.M.A.N. könnte man dank BLA mehrere Instanzen über einen normalen (realen oder virtuellen) L2-Switch zusammenschalten. Selbst ohne Metrikübermittlung wäre dann das Routing zwischen den Instanzen optimal, muss aber immer über den Switch laufen. Bei dieser Konstruktion teilt man das Routing des Netzes in zwei oder mehr Bereiche auf, sodass der Routingtraffic in jeder Instanz nur mit der Größe der Instanz wächst, nicht mit der Größe des gesamten Netzes. Allerdings erscheinen auf den Routern im Innern der Instanz in den B.A.T.M.A.N. Tabellen alle MAC Adressen aus den anderen Instanzen als clients der Router am Switch ("externe Routen"). Mglw. könnte so eine Kopplung auch durch die Präsenz zweier Instanzen innerhalb eines Routers erfolgen: "As of 2010.2.0 it is possible to let a single mesh node participate in mutliple mesh clouds at the same time which makes it necessary to assign interfaces to individual mesh clouds and having multiple batX interfaces." (B.A.T.M.A.N. Doku)

Fazit: Mglw. könnte man mit BLA größere B.A.T.M.A.N. Netze bauen als mit einer einzelnen Instanz. Aber wer bitteschön will denn noch größere Broadcastdomänen, wie sie hierbei - mit Hilfe des Switches - zustandekämen? Allerdings besteht ein interessanter Aspekt des Routings auf Layer 2 darin, dass interessierte Freifunkerinnen unmittelbar eine Verbindung auf Layer 2 zueinander aufbauen können, was die Nutzungsmöglichkeiten einer freien Netzwerkinfrastruktur erweitert. Doch dafür müssen wohl andere Lösungen untersucht werden, als die Ausdehnung des L2 Routing auf das FF-Kabelnetz.)

Anmerkung: Die Begriffe “Autonomes System” und “externe Route”

Weil die Verwendung mehrerer, gleichrangiger IGPs innerhalb eines Netzes für die Leserin mglw. ungewohnt ist, sei hier betont, dass in diesem Text mit dem Begriff “externe Route” nicht (nur) solche Routen gemeint sind, die aus einem anderen AS (AS im Sinne eines EGP) stammen, sondern alle Routen, die nicht aus dem betrachteten IGP stammen. “Extern” sind hier also auch alle Routen, die aus einem benachbarten IGP kommen, und nicht nur die per eBGP gelernten. “AS” hat bei IGP eben eine andere Bedeutung als bei EGP. So heißt es in der OSPF Spezifikation (RFC 2328):

```
Autonomous System
   A group of routers exchanging routing information via a
   common routing protocol. Abbreviated as AS.
```

Dieser Text hier verwendet dafür stattdessen den Ausdruck “IGP Instanz”, um den Begriff “AS” eindeutig verwenden zu können, d.h. den EGPs zu überlassen. Beim Lesen der noch folgenden Zitate aus der OSPF Spezifikation also bitte dran denken, das “AS” aus dem Zitat mit “IGP Instanz” zu übersetzen.

Ein Konzept für die Kopplung von IGP Instanzen

Im Kontext des Fußgängerzonenproblems besteht die wesentliche Motivation für einen Verbund verschiedener IGPs in einem (FF-)Netz darin, das Fluten / die Flut von Routinginformationen zu minimieren, und trotzdem im ganzen Netz optimales Routing zu erzielen, wie man es innerhalb eines einzelnen IGPs kennt. Die zentrale Gleichung dieses Ansatzes lautet:

```
Kopplung von IGPs = gemeinsame Metrik trotz Entkopplung der Topologien
```

Das ist übrigens nichts Neues, sondern wird schon in der OSPF Spezifikation formuliert. (Zitat s.u. im Abschnitt “OSPF areas”).

Leider kommt OSPF selbst nicht als (einheitliche) Lösung für FF-Netze in Frage, und zwar wegen seiner Unbrauchbarkeit in WLAN adhoc Meshes (s.o.). Wie aber ein Netz aufgebaut werden kann, dass diese Idee nicht durch areas innerhalb einer OSPF Instanz realisiert, sondern mit mehreren eigenständigen IGP Instanzen, soll weiter unten noch untersucht werden.

Globale Metrik auch bei unterschiedlichen IGPs?

Aus der gewünschten Metriktreue (zwecks global optimalem Routing) folgt nicht zwingend die Verwendung genau gleicher Metriken innerhalb der beteiligten IGP Instanzen. Diese müssen nur an den Grenzen der IGPs in eine globale (gemeinsame) Metrik umgerechnet werden, wobei die strenge Monotonie der jeweiligen Metriken erhalten bleibt. Der Einfachheit halber lassen wir diesen Schritt im Folgenden weg und gehen von gleichbedeutenden (und praktischerweise arithmetischen) Metrik-Werten in allen beteiligten IGPs aus. (Zu allgemeineren Metriken lohnt es sich, die bereits erwähnte Babel RFC zu goutieren.)

Vor dem Weiterlesen sollte sich der Leser nochmal vergegenwärtigen, dass optimales Routing zwischen Routingdomänen nicht auf die Topologieinformationen aus den verschiedenen Routingdomänen angewiesen ist, sondern nur auf deren Metriken, die als externe Routinginformation in das jeweilige IGP exportiert (“redistributed”) werden. (S.o. die “zentrale Gleichung” und das OSPF-Zitat.)

So machen’s die Großen[tm]

BGP - MED

Die offensichtliche und übliche Möglichkeit zur Verbindung von IGPs ist die Verwendung eines EGP (d.h. eBGP). Im FF-Kontext würde also über den Uplink zwischen Straßenrouter und VPN-Endpunkt am besten eBGP gesprochen werden, um das IGP des Mesh mit dem IGP des Backbones zu verbinden. (iBGP ginge auch, aber führt hier wg. des “i” nur zu Verwirrung.) Die Metriken können bei BGP über das MED Attribut der BGP Routen zwischen den verschiedenen IGPs transportiert werden. (MED = Multi-Exit Discriminator, s.a. RFC 4451.)

Nachteil dabei ist,

- dass BGP nicht gerade für die FF-straßenüblichen (Konsumenten-)Uplinks gedacht war;-) S. zB route flap dampening bei überlasteten Uplinks.

- Außerdem tauscht BGP seine Routinginformationen über TCP aus, kann also nur eine konfigurierte Liste von Nachbarn ansprechen, sie aber nicht per Broadcast oder Multicast "adhoc" finden. Das bedeutet im FF-Szenario, dass die VPN-Endpunkte ebenso viele BGP-peers konfigurieren müssten, wie sie VPN clients (Konsumentenuplinks) bzw. peers (zB IC-VPN) versorgen. Da BGP ein vergleichsweise aufwändiges Protokoll ist, bedeutet diese Lösung aus dem Kontext der Großen[tm] aber im FF-Kontext eher eine Belastung der "Supernodes" (von der Belastung der Konfigurationsmechanik und der Plasterouter mal zu schweigen;).

Zusammenfassung: BGP-MED ist eine echte Option, kann aber in der Praxis Probleme machen, insb. beim Skalieren.

OSPF areas

OSPF erreicht seine Reduktion des Routingtraffics auf zwei Ebenen. Zunächst dadurch, dass OSPF ein link-state Protokoll ist. Dessen Kern ist die link-state Datenbank (LS DB), die auf jedem Router der betrachteten area den gleichen Stand haben muss. Nur unter dieser Voraussetzung kann jeder Router autonom den SPF-Baum mit den kürzesten Pfaden zwischen ihm und allen anderen Routern berechnen, so dass die von den verschiedenen Routern getroffenen Routingentscheidungen zu einem schleifenfreien und optimalen Routing führen. Der Umfang der RFC 2328 (240 Seiten) rührt auch von der Komplexität, die Konsistenz der LS DBs auf den verschiedenen Routern unbedingt sicherzustellen - und zwar mit möglichst wenig Routingtraffic.

Hinzu kommt das Konzept des Designated Routers (DR) in jedem Subnetz der area, das eine broadcast (oder zumindest NBMA) Topologie hat:

```
Designated Router
        Each broadcast and NBMA network that has at least two
        attached routers has a Designated Router.  The Designated
        Router generates an LSA for the network [...]
        The Designated Router concept enables a reduction in the
        number of adjacencies required on a broadcast or NBMA
        network.  This in turn reduces the amount of routing
        protocol traffic and the size of the link-state database.
```

Wie wichtig solch ein "Detail" für größere Netze sein kann, lässt sich auch an der oben erwähnten Formel r * log(r) + v ablesen. Das v steht darin für die Anzahl der direkten Verbindungen zwischen Routern. In einem broadcast (oder NBMA) Subnetz mit r' Routern, die ein interface in dieses Subnetz haben, wächst die Verbindungsanzahl des Subnetzes (v') quadratisch mit der Anzahl der Router:

v' = 1 + 2 + 3 + ... + (r' - 1) = r' * (r' - 1) / 2

(Anmerkung: OLSR ist auch ein link-state Protokoll, und dort lässt sich ein zum DR ähnliches Konzept entdecken: multipoint relay (MPR).)

Die zweite Ebene der Reduktion von Routingtraffic besteht darin, die OSPF Instanz (in OSPF-Sprech: das "AS") in areas aufzuteilen, die jeweils eigene, unabhängige LS DBs haben. Wenn man zB eine OSPF Instanz in a gleich große areas aufteilt, ergibt sich als durchschnittliche Berechnungskomplexität für jeden Router:

r/a * log(r/a) + k/a = r/a * (log(r) - log(a)) + k/a

D.h. schon eine Aufteilung in zwei areas kann bei größeren Netzen den (Berechnungs-)Aufwand spürbar verringern. Ähnlich folgt aus der Halbierung des Konnektivitätsgraphen - und damit der LS DB - die entsprechende Reduktion des Routingtraffics, der für die Synchronisation der LS DBs auf den Routern innerhalb einer area anfällt.

Die Routinginformationen aus der anderen area tauchen nur als area-externe Routen auf. D.h. wenn sich eine solche Route ändert, so muss jeder Router der betrachteten area *nicht* den SPF Baum neu berechnen - sondern nur die zu dieser Route gehörige Zeile in der Routingtabelle (RIB) ändern. Für Metrikänderungen braucht es dazu nur eine Addition (neue Metrik = neue externe Metrik + unveränderte SPF Distanz).

Zusammenfassung: OSPF areas sind ein optimierter Kandidat um größere Netze so zu strukturieren, dass der Routingtraffic minimiert wird. Doch leider taugt OSPF nicht für WLAN adhoc Meshes:-(

And the winner is ...

Beide Varianten aus der Welt der Großen führen jeweils für sich alleine nicht zum FF-Glücklichrouting. Aber in einem üblicherweise heterogenen FF-Netz gibt es ja verschiedene Problemzonen, die mit verschiedenem Werkzeug bearbeitet werden könnten und sollten:

- Im WLAN adhoc Mesh braucht es auf jeden Fall ein darauf angepasstes Protokoll (OLSR, B.A.T.M.A.N., Babel, ...).
- Im stabilen Teil innerhalb des AS (größere Uplinks, VPN peering Netze, Richtfunknetze) kann OSPF für minimalen Routingtraffic sowie schnelle Konvergenz sorgen.
- Und nach außen schließlich, beim Austausch mit dem "echten" Internet muss man sowieso BGP sprechen. (Evtl. auch schon auf den Uplinks, s.o. BGP-MED.)

Zusammenfassung: the winner is ... die Integration verschiedener Routingprotokolle unter Verwendung einer globalen Metrik. (Mit "global" ist hier (noch;) nicht die ganze Welt(herrschaft;;) gemeint, sondern alle Teile eines (heterogenen) FF-Netzes, die Routinginformationen (und ergo Datenpakete) direkt mit mindestens einem anderen Teil des betrachteten FF-Netzes austauschen.

So geht's also auch: mit einer open source routing suite

Mit "routing suite" ist eine Software gemeint, die mehrere Routingprotokolle implementiert und integriert. Inkusive der FIB(s) des Kernels, der ja die eigentliche Arbeit der Paketweiterleitung ausführt. Open Source Beispiele sind Quagga, XORP und Bird. Mit einer solchen Routingsoftware können IGPs verbunden werden, ohne dass ein EGP gebraucht wird. Die Übertragung der Routinginformationen (insb. Metrik) erfolgt über die internen Mechanismen der Suite, also Routingtabellen und die Filterung und Manipulation von Routen. (Letzteres wäre zB der Ort für die oben erwähnte Umrechnung von Metriken.)

Eine Verbindung von zwei IGP Instanzen geschieht also im Inneren eines Routers, der an beiden Instanzen teilnimmt. In seine zentrale Routingtabelle kommen die besten Routen aus beiden IGPs. Von diesen Routen wiederum wird die mit der besten (globalen) Metrik in die FIB eingetragen. Diese Route (the best of the best:) wird nun jeweils in das IGP redistribuiert, aus dem sie nicht stammt. D.h. ein solcher Grenzrouter verwendet die global beste Route zu einem Ziel, und verkündet sich als Quelle dieser Route (mit globaler Metrik) in das IGP, das (vom Grenzrouter aus gesehen) noch keinen besseren Weg zum Ziel dieser Route kennt.

So trivial dieser Ansatz erscheint - so wenig scheint er realisiert worden zu sein. Natürlich werden im FF routing suites verwendet, aber üblicherweise "nur" für BGP und gelegentlich OSPF. Es bleibt zu hoffen, dass sich das Blatt zB mit der o.g. Integration von Babel in Bird wendet, weil diese Idee dadurch für die "kleinen" FF-Netzwerke in praktikable Reichweite kommt und dank Babel endlich auch zur metriktreuen Anbindung von WLAN adhoc Meshes angewendet werden kann. Die Großen[tm] hingegen werden sich für diese Idee wohl kaum interessieren, weil für deren Routingstrategien bzgl. externer Routen solche routing policies, die den cash flow optimieren, wichtiger sind als (technisch) optimale Methoden. (Und das heißt eBGP nach außen und OSPF (oder IS-IS) und/oder iBGP nach innen. Kein Wunder also, dass dies (noch?) die Kernausstattung auch der freien routing suites darstellt.)

Auch wenn das Fußgängerzonenproblem mit 2 IGPs gelöst werden kann, muss man dabei nicht stehenbleiben. Ein "echtes" AS (im Sinne von EGP) wie das AS201701 des FF-Rheinland umfasst auch die Subnetze vieler Communities, die ihrerseits (mindestens) eine IGP Instanz betreiben. Diese werden zZ (wie bei den Großen[tm]) per BGP miteinander und mit den Borderroutern des AS verbunden. Wie man hörte, ist das dadurch resultierende Routing nicht optimal. Das ließe sich durch (mindestens:) 1 (link-state) IGP zwischen Supernodes bzw./und Borderroutern sicherlich verbessern. Supernodes der Communities könnten untereinander optimal routen.

Weitergehend könnten zB Konsumentenuplinks eines Accessproviders auf einem Router mit Peering zu diesem Accessprovider terminiert werden, und verschiedene IGP Instanzen (zB WLAN adhoc Meshes und was sonst so an den Uplinkroutern an der Straße dranhängt, etwa Hausnetze und Server) optimal verbinden, ohne dass der Backbone belastet wird. Klare Sache: je weniger der FF als default route Provider (Datenautobahnzubringer) fungiert, d.h. je mehr im FF-Netz selbst passiert, umso interessanter werden Routingmethoden, die auch ohne Backbone funktionieren und skalieren.

Was ist der Unterschied zwischen IGP-Kopplung und OSPF areas?

Wenn man sich das Konzept der IGP-Kopplung als Variante von OSPF (oder IS-IS) areas vorstellt, dann fällt genau eines auf: bei der IGP-Kopplung fehlt die backbone area;-) Aufgabe dieser "area 0" ist es bei OSPF, alle Grenzrouter miteinander zu verbinden, d.h. Grenzrouter zwischen areas sind bei OSPF immer auch Teil der area 0. Dadurch besteht der Weg eines Pakets innerhalb einer (in areas gegliederten) OSPF Instanz konzeptionell aus drei Teilen:

- zunächst der Weg innerhalb der area, in der das Paket losgeschickt wurde, bis zu einem Grenzrouter dieser Start-area,
- dann der Weg durch die area 0 bis zu einem Grenzrouter der Ziel-area,

- und schließlich der Weg von diesem Grenzrouter durch die Ziel-area zum Ziel.

Klingt übersichtlich - aber muss das so gemacht werden? Schauen wir ins Original:

Die OSPF backbone area

```
3.1. The backbone of the Autonomous System

     The OSPF backbone is the special OSPF Area 0 (often written as
     Area 0.0.0.0, since OSPF Area ID's are typically formatted as IP
     addresses).  The OSPF backbone always contains all area border
     routers.  The backbone is responsible for distributing routing
     information between non-backbone areas. The backbone must be
     contiguous.  However, it need not be physically contiguous;
     backbone connectivity can be established/maintained through the
     configuration of virtual links.

     Virtual links can be configured between any two backbone routers
     that have an interface to a common non-backbone area.  Virtual
     links belong to the backbone.  The protocol treats two routers
     joined by a virtual link as if they were connected by an
     unnumbered point-to-point backbone network.  On the graph of the
     backbone, two such routers are joined by arcs whose costs are
     the intra-area distances between the two routers.  The routing
     protocol traffic that flows along the virtual link uses intra-
     area routing only.
```

Wenn die Routinginformationen beim Transit durch eine area (d.h. über einen virtual link) die innerhalb dieser area geltende Metrik beibehalten, wo ist dann der Unterschied dazu, die area ohne virtual link als Transit zu benutzen? So heißt es dann auch kurz darauf im RFC:

```
3.2. Inter-area routing

     [...]
     The correct area border router to use as the packet exits the
     source area is chosen in exactly the same way routers
     advertising external routes are chosen.
```

backbone area und Schleifenfreiheit

Warum kann man dann nicht gleich die Routen von anderen areas als externe Routen behandeln, und auf das Konzept backbone area verzichten? Dafür könnte es auf Anhieb zwei Gründe geben. Der wichtige Grund ist, dass die backbone area unverzichtbar ist, insb. um die Schleifenfreiheit zu gewährleisten. Da das routing innerhalb jeder area aber optimal ist (dank OSPF:-), ist nicht klar, wo globale Schleifen entstehen sollen, wenn jeder Teil des Ganzen sich optimal verhält und den Rest des Ganzen einfach als “extern” ansieht. Sucht man in der OSPF RFC nach “loop” findet man (außer “loopback”:) nur Folgendes:

```
    Before the lookup begins, "discard" routing table entries should
    be inserted into the routing table for each of the router's
    active area address ranges (see Section 3.5).  (An area range is
    considered "active" if the range contains one or more networks
    reachable by intra-area paths.)  The destination of a "discard"
    entry is the set of addresses described by its associated active
    area address range, and the path type of each "discard" entry is
    set to "inter-area".[10]
[...]
[10] "Discard" entries are necessary to ensure that route
summarization at area boundaries will not cause packet looping.
```

Das ist sicherlich ein Thema, das man bei jeder Anwendung von Routenaggregation mitbedenken muss. Aber das ist keine Antwort auf die Frage nach der Unverzichtbarkeit einer backbone area. Aus der Sicht der OSPF Spezifikation

stellt sich die Frage vmtl. deshalb nicht, weil die Spezifikation durch das Konzept backbone area das Schleifenproblem ja gelöst hat:

```
3.2. Inter-area routing

     When routing a packet between two non-backbone areas the
     backbone is used.  The path that the packet will travel can be
     broken up into three contiguous pieces: an intra-area path from
     the source to an area border router, a backbone path between the
     source and destination areas, and then another intra-area path
     to the destination.  The algorithm finds the set of such paths
     that have the smallest cost.

     Looking at this another way, inter-area routing can be pictured
     as forcing a star configuration on the Autonomous System, with
     the backbone as hub and each of the non-backbone areas as
     spokes.
```

Und in einer Sterntopologie kriegt auch der Fortgeschrittene keine Schleife gebaut;-) OK, wir wissen also, dass OSPF mit backbone area schleifenfrei ist. Theoretisch und praktisch. Logisch betrachtet handelt es sich also bei der backbone area um eine hinreichende Bedingung für Schleifenfreiheit: backbone area => schleifenfrei. D.h. die Entscheidung der Spezifikation für das Konzept backbone area schließt noch nicht aus, dass es auch ohne backbone area geht. Hinreichend ist eben etwas anderes als notwendig.

backbone area als Panopticon

Der zweite Grund für eine backbone area ist eine mögliche praktische Relevanz der folgenden Bestimmung: Die backbone area ist der logische Ort einer Gesamtschau über die Instanz, also über alle areas. Hub and spoke hat daher auch eine Analogie zu Bentham's Panopticon, das ja durch die jüngere Kybernetikrezeption (S. 14) und erst recht Snowden etc. wieder ins Bewusstsein der Massen gelangte.

Bei der IGP-Kopplung hingegen gibt es diesen logischen Ort einer Gesamtschau nicht mehr: jede der beteiligten IGP-Instanzen kennt nur noch die Richtung (des ersten Teilstücks des Weges) und die Länge des Weges, aber mehr nicht. Im Gegensatz dazu ist die panoptische backbone area ja Träger des gesamten inter-area traffics, und hat daher sowohl einen Überblick über die Struktur des Gesamtgeschehens als auch für jeden Weg Kenntnis der Grenzübertritte (des Pakets von einer area in eine andere).

Für die operative Praxis wird der obige Reflexionsüberschuss wohl weniger interessieren als der evtl. Nutzen eines Gesamtüberblicks zur Hege und Pflege des Routinggeschehens in einem Netz, das in areas oder gekoppelte IGPs strukturiert ist. Dennoch führt uns die eher sozialphilosophische Betrachtung des vorigen Absatzes zu einem Lösungansatz für die immer noch offene Frage, ob eine backbone area notwendig ist. Und zwar deshalb, weil die obige Formulierung "jede der beteiligten IGP-Instanzen kennt nur noch die Richtung (des ersten Teilstücks des Weges) und die Länge des Weges" überdeutlich nach DV Routing klingt: "Distance-Vector-Protokolle bestimmen die Erreichbarkeit durch einen Vektor aus Entfernung und Richtung." (Wikipedia)

Entspricht IGP-Kopplung einem DV Protokoll?

Wenn man versucht, IGP-Kopplung als DV Routing zu sehen, stellt sich (insb. dank der früher leider oft leidigen Erfahrungen mit R.I.P.) sofort die Frage, wie denn Schleifenfreiheit garantiert werden kann? Und das ganz ohne einen Gesamtüberblick, wie ihn die link-state DB innerhalb einer area und die backbone area zwischen den areas darstellen?

Bedienen wir uns auf der Suche nach einer Antwort doch erstmal wieder bei den Großen[tm], genauer gesagt beim Allergößten, beim Internet (also dem internet mit großem "I"). Dessen globales Routing wird durch ein DV Protokoll organisiert, nämlich BGP. Eine wesentliche Komponente der Routinginformation von BGP besteht aus dem Pfad aller von der Route durchquerten ASe, sodass man bei BGP auch von einem Pfad-Vektor Protokoll (PV) spricht. Aufgrund des Datenaufkommens in Kernbereichen des Internet würde eine dort entstehende Routingschleife entsprechend große Auswirkungen haben. Die alltägliche Erfahrung globaler Connectivity ist also ein praktischer Hinweis darauf, dass BGP einen guten Job macht. Dabei sollte man natürlich die Bedeutung der Arbeit aller BGP Admins für die Stabilität des globalen Internet nicht vergessen. Aber die Genannten können aus dem Protokoll nur das rausholen, was drinsteckt. Also: wie kriegt man globale Schleifenfreiheit hin? BGP muss eine wirksame Antwort enthalten, und deshalb können wir ohne Kenntnis dieser Antwort schon jetzt deduzieren: Von BGP lernen heißt siegen lernen. [SCNR] Die Antwort selbst wurde freundlicherweise auch schon andernorts (RFC 6126 (Babel), 2.4) aufgeschrieben:

```
For example, BGP
can be modelled as being a distance-vector protocol with a (rather
drastic) feasibility condition: a routing update is only accepted
when the receiving node's AS number is not included in the update's
AS-Path attribute (note that BGP's feasibility condition does not
ensure the absence of transitory "micro-loops" during reconvergence).
```

Bei der IGP-Kopplung gibt es zwar keinen Pfad (der durchquerten IGP-Instanzen), sehr wohl aber eine globale Metrik. Anhand derer kann sich jeder beteiligte Router für den global optimalen Weg entscheiden, eben den mit der geringsten globalen Metrik.

Idee: Die Optimalität der Routingentscheidung an den Grenzroutern führt zur Transitivität der Optimalität der Entscheidung über Routen innerhalb der IGPs. Die Optimalität der Grenzrouterentscheidungen beruht technisch auf dem Verfahren, die Routen in die zentrale Routingtabelle einzutragen (und die zentralen Routen ggf. aus dieser Tabelle heraus in die anderen IGPs, an denen der Grenzrouter teilnimmt, zu exportieren). Zitat von oben (s. routing suite):

- Eine Verbindung von zwei IGP Instanzen geschieht also im Inneren eines Routers, der an beiden Instanzen teilnimmt.
- In seine zentrale Routingtabelle kommen die besten Routen aus beiden IGPs.
- Von diesen Routen wiederum wird die mit der besten (globalen) Metrik in die FIB eingetragen.
- Diese Route (the best of the best:) wird nun jeweils in das IGP redistribuiert, aus dem sie nicht stammt.
- D.h. ein solcher Grenzrouter verwendet die global beste Route zu einem Ziel, und verkündet sich als Quelle dieser Route (mit globaler Metrik) in das IGP, das (vom Grenzrouter aus gesehen) noch keinen besseren Weg zum Ziel dieser Route kennt.

Dadurch kann man den Prozess der Verbreitung der Route im Verbund der IGPs aus der Sicht des Routers, von dem die Route ursprünglich stammt, so beschreiben:

- Die Nachbarrouter des Quellrouters erhalten von diesem die (neue) Route und tragen sie in die zentrale RIB (und die FIB) ein. Der erste Schritt sieht also aus wie hub and spoke.
- Im nächsten Schritt verbreiten die Nachbarn des Quellrouters die Route an ihre Nachbarn. In diesem (und allen weiteren) Schritten kann es passieren, dass ein Router die Route mehr als einmal erhält. Er entscheidet sich dann anhand der jeweiligen globalen Metrik für die Route, die auf dem kürzesten Weg gekommen ist. Jedenfalls trägt er (modulo Verallgemeinerungen wie ECMP) genau eine Route in die FIB ein, und zwar eine, die faktisch realisierbar ist, da in der Reihe der Vorgängerrouter auf dem Weg der Route von der Quelle bis zum gerade betrachteten Router, jeder Router genau einen "uplink next hop" mit (aus dessen Sicht) minimaler Entfernung von der Quelle unter allen Nachbarn hatte, und die globale Metrik der Route entlang dieses Weges streng monoton sinkt, also letztlich bei 0 ankommen muss, also den Quellrouter erreicht.

Faktische Realisierbarkeit und Eindeutigkeit des Weges über den (laut globaler Metrik) besten next hop führen bei jedem Schritt der Verbeitung der Route zu einer Baumstruktur, deren Wurzel der Quellrouter der betrachteten Route ist. Eine Baumstruktur ist schleifenfrei, q.e.d.

Doch handelt es sich beim zweiten Blick auf den vorigen Absatz dort nicht um den Bellman-Ford Algorithmus, den Standard für DV? Und so lesen wir erschüttert im unmittelbar folgenden Abschnitt der gerade zitierten Babel RFC:

```
2.3. Transient Loops in Bellman-Ford

   It is well known that a naive application of Bellman-Ford to
   distributed routing can cause transient loops after a topology
   change.  Consider for example the following diagram:

            B
         1 /|
      1   / |
   S --- A  |1
          \ |
         1 \|
            C
```

```
After convergence, D(B) = D(C) = 2, with NH(B) = NH(C) = A.

Suppose now that the link between S and A fails:

          B
       1 /|
        / |
S      A  |1
        \ |
       1 \|
          C

When it detects the failure of the link, A switches its next hop to B
(which is still advertising a route to S with metric 2), and
advertises a metric equal to 3, and then advertises a new route with
metric 3.  This process of nodes changing selected neighbours and
increasing their metric continues until the advertised metric reaches
"infinity", a value larger than all the metrics that the routing
protocol is able to carry.
```

counting-to-infinity :-(

Der Effekt nennt sich counting-to-infinity. Und den will man nicht. Aber kann das wirklich auch bei IGP-Kopplung passieren? Weiter oben hieß es doch: “Eine Baumstruktur ist schleifenfrei, q.e.d.”

Wesentlicher Bestandteil der Beweisskizze war aber die faktische Realisierbarkeit der Route, d.h. dass sie einen Weg weist, auf dem die Datenpakete den Zielpräfix am Quellrouter wirklich erreichen können. Bei einer Partitionierung des IGP-Verbindungsgraphen gilt das aber nicht mehr, wie man am obigen Beispiel aus der Babel RFC sehen kann. Der Partionierung zwischen den Routern { S } und { A B C } führt dazu, dass die drei abgetrennten Router für die Route mit Quelle S keine Routingentscheidungen mehr treffen können, die sich an faktisch realisierbaren Wegen (und ihrer Länge) orientieren.

So weit, so schlecht. Dafür haben wir immerhin eine Arbeitshypothese zum Verständnis der IGP-Kopplung:

```
IGP-Kopplung = DV mit naivem Bellman-Ford Algorithmus
```

Wenn man nun den counting-to-infinity Effekt in einer IGP-Kopplung beobachten würde, wäre das eine Stütze für die Arbeitshypothese, weil diese dann eine richtige Vorhersage gemacht hat. Zur Veranschaulichung des counting-to-infinity Effekts rechnen wir das obige Beispiel aus der Babel RFC also einmal als IGP-Kopplung durch. Dazu gibt es einen Anhang in ASCII art, um hier den Gedankenfluss nicht zu splattern. Wer sich für die Mechanik dieser IGP-Kopplung oder allgemeiner für die Möglichkeiten einer routing suite interessiert, dem sei der Anhang jedenfalls empfohlen.

Was man dort auch recht genau sehen kann und was hier noch interessiert, ist die Beobachtung, dass der Kreislauf der Route (T=5 ... T=8) nicht zu einem Kreislauf der Datenpakete führt. Das liegt daran, dass die Route immer nur auf einem der drei am Kreislauf beteiligten Router in der zentralen RIB und damit der dortigen FIB eingetragen ist. Würden die anderen beiden Router ein Datenpaket mit Ziel S erhalten, würden sie es verwerfen (“destination unreachable”). Durchschnittlich ein Drittel aller Datenpakete Richtung S werden also (maximal) einen Schritt (= 1 IGP-Instanz) in dem Kreis zurücklegen, ehe sie vom nächsten Grenzrouter verworfen werden - die anderen zwei Drittel werden sofort verworfen. Daher ist counting-to-infinity für die Praxis längst nicht so fatal wie eine “echte” Schleife, in der alle vom Kreis eingefangenen Datenpakete bis zum Ablauf ihrer TTL unterwegs sind. Die Belastung der am Kreis beteiligten Router (incl. derer innerhalb der IGP-Instanzen) durch die ständigen Updates bei counting-to-infinity steht allerdings auf einem anderen Blatt.

backbone area rettet !-)

Nebenbei haben wir jetzt noch ein handfestes Argument für die backbone area gefunden, denn soviel ist klar: counting-to-infinity wäre mit OSPF nicht passiert. Und das ist ein anderes Kaliber an Argument als die bisher weiter oben aufgeführten, insb. als das praktische bzw. sozialphilosophische Argument (Gesamtüberblick über das Routinggeschehen im Netz). Zum Argument der Verhinderung von “echten” Schleifen dank backbone area ist allerdings zu ergänzen, dass (naives) Bellman-Ford auch Schleifenfreiheit (bei stabiler Topologie) herstellt (s. Beweisskizze und Literatur und vor allem die Verbreitung von Bellman-Ford im Bereich des DV-Routing - FF-relevant sind akut B.A.T.M.A.N. und Babel).

feasibility conditions

Im Lichte der Arbeitshypothese muss die IGP-Kopplung aber möglicherweise noch ein anderes Problem aus der Babel RFC wegstecken. Um das zu verstehen, lesen wir erstmal weiter in der RFC:

```
Bellman-Ford is a very robust algorithm: its convergence properties
are preserved when routers delay route acquisition or when they
discard some updates.  Babel routers discard received route
announcements unless they can prove that accepting them cannot
possibly cause a routing loop.

More formally, we define a condition over route announcements, known
as the feasibility condition, that guarantees the absence of routing
loops whenever all routers ignore route updates that do not satisfy
the feasibility condition.  In effect, this makes Bellman-Ford into a
family of routing algorithms, parameterised by the feasibility
condition.
```

Uff: Router die Beweise führen - das klingt zumindest nicht nach "naivem" Bellman-Ford;-) Und WTF ist eine "feasibility condition"? Da hilft ein Re-Zitat von oben:

```
Many different feasibility conditions are possible.  For example, BGP
can be modelled as being a distance-vector protocol with a (rather
drastic) feasibility condition: a routing update is only accepted
when the receiving node's AS number is not included in the update's
AS-Path attribute [...]
```

OK: diese feasibility condition ist also ein Schlüssel zur globalen Schleifenfreiheit des Internet (mit großem "I"). Babel ist aber nicht BGP, und hat vmtl. ebensowenig wie die IGP-Kopplung die Routing-Weltherrschaft im Sinn (die schließlich in weltweit optimalem Routing gipfeln würde:). Jedenfalls benutzt Babel eine (vom proprietären EIGRP inspirierte) feasibility condition, mit der die o.g. "Beweise" zu einfachen Metrikvergleichen werden. Dazu formuliert Babel seine feasibility condition mit Hilfe der feasibility distance = die kleinste Metrik, die der Router bisher für das betrachtete Ziel an seine Nachbarn weitergegeben hat.

```
Given a router A, define the feasibility distance of
A, written FD(A), as the smallest metric that A has ever advertised
for S to any of its neighbours.  An update sent by a neighbour B of A
is feasible when the metric D(B) advertised by B is strictly smaller
than A's feasibility distance, i.e., when D(B) < FD(A).
```

Diese feasibility condition führt dazu, dass der Router auch solche Routen annimmt, die schlechter sind als die aktuell beste, aber ebenso schleifenfrei. Dadurch kann der Router bei Verschlechterung oder Verschwinden der besten Route sofort eine Ersatzroute aktivieren, ohne dadurch Schleifen zu erzeugen. Dieser wichtige praktische Vorteil schneller Rekonvergenz à la LS hat aber einen Preis: route starvation (die es bei LS nicht gibt).

route starvation ?

```
Obviously, the feasibility conditions defined above cause starvation
when a router runs out of feasible routes.
```

Da die feasibility distance per Definiton streng monoton fallend ist, kann eine Situation eintreten, in welcher der Router zwar noch eine Route für das betrachtete Ziel von mindestens einem Nachbarn empfängt, diese aber nicht annimmt, weil sie eine größere Metrik hat als die feasibility distance. D.h. Datenpakete für das Ziel werden von diesem Babel-Router verworfen, obwohl das Ziel noch erreichbar ist.

Das Problem wird bei Babel et al. durch ein streng monoton steigendes Attribut der Routen gelöst, die Sequenznummer. Wenn ein Router merkt, dass er keine funktionierende Route mehr wg. seiner feasibility distance annehmen kann, erwirkt er bei der Quelle der Route ein Inkrement der Sequenznummer, sodass der SPF-Baum für diese (neue) Route neu aufgebaut wird, d.h. auch der "abgehängte" Router sich wieder für einen optimalen Weg entscheiden kann, da eine (neue) Route mit höherer Sequenznummer per Definition immer eine bessere Metrik hat als eine (alte) Route mit niedrigerer Sequenznummer - unabhängig von der Weglänge:

```
A received update is feasible when either it is more recent than the
feasibility distance maintained by the receiving node, or it is
equally recent and the metric is strictly smaller.  More formally, if
FD(A) = (s, m), then an update carrying the distance (s', m') is
feasible when either s' > s, or s = s' and m' < m.
```

Spannende Sache, aber das hilft der IGP-Kopplung nicht erkennbar auf die Beine, da diese ja nur aus ein bisschen Mechanik in der routing suite besteht, welche sich ihrerseits nur auf die gemeinsamen Attribute der in der Kopplung benutzten IGPs abstützen kann. Oder anders gesagt: selbst wenn Babel in Bird integriert wird, nutzt die pfiffige Methode von Babel (feasible distance + Sequenzierung der Routen) der Kopplung als ganzer nix. Sie nutzt nur was in den IGP-Instanzen der Kopplung, die Babel sprechen.

Die lokale Eigenschaft schneller (Re-)Konvergenz in einer beteiligten IGP-Instanz ist zweifelsohne wichtig für das (Re-)Konvergenzverhalten auf der ("globalen") Ebene der Kopplung - kann diese aber nicht ersetzen. Die innerhalb eines Routingprotokolls kodierte Pfiffigkeit ist (de facto) nicht unmittelbar auf der Ebene der Mechanik der routing suite nutzbar, die ja "nur" dafür da ist, sich um die Integration der implementierten Protokolle zu kümmern.

So weit, so schlecht. Aber die Optimistin kann hier auch eine gute Nachricht aus dem Routing-Kaffesatz herauslesen: wenn wir keine (erkennbare) Möglichkeit haben, das Konzept "feasibility distance" auf der Ebene der routing suite, d.h. auf der Ebene des impliziten DV Routings zwischen den Instanzen ("inter-area") zu implementieren, dann bleibt uns vmtl. auch die route starvation erspart.

Beim jetzigen Stand des Diskurses ist die IGP-Kopplung als naive Bellman-Ford Implementierung anzusehen, die dieses Problem nicht hat - aber dafür counting-to-infinity. Ein Mangel an Alternativrouten ist jedenfalls kein Problem, denn die sind bei IGP-Kopplung ausreichend vorhanden: aus jeder IGP-Instanz, an der ein Grenzrouter teilnimmt, kommt die (aus Sicht des betrachteten Grenzrouters) beste Route, welche er aus der betrachteten IGP-Instanz empfängt, in die zentrale RIB. Und das ist solange der Fall, wie die vom betrachteten Router in eine IGP-Instanz redistribuierte Route für das Ziel nicht besser ist als alle Routen, die alle anderen Grenzrouter dieser Instanz aus anderen Instanzen bekommen haben. (Dabei ist die Weglänge durch das IGP mitzuberechnen ... aber wir vertiefen das hier (noch) nicht, sondern kümmern uns erstmal um die bisherigen Ergebnisse zum Thema "IGP-Kopplung als DV Routing".)

Zusammenfassung: IGP-Kopplung scheint global wie ein DV Routing zu funktionieren. In seiner bisher diskutierten naiven Form unterliegt IGP-Kopplung daher dem counting-to-infinity Problem, aber nicht der route starvation. Es reicht als naives DV Protokoll aber nicht an Babel, OSPF oder BGP heran. Die offene Frage nach einer notwendigen Bedingung für global optimales Routing lässt sich also mit gutem Bauchgefühl im Bermudadreieck zwischen naivem Bellman-Ford und backbone area bzw. BGP an den anderen Ecken des Dreicks vermuten.

```
backbone area ------------- BGP     hinreichend
           \                /
            \      ?       /       notwendig
             \            /
          Bellman-Ford (naiv)       zu schwach (counting-to-infinity)
```

Laborstudie: Kopplung von konvexen IGP-Instanzen mit kompatibler Metrik zu einer gemeinsamen Routingdomäne mit global kürzesten Wegen und Toleranz gegen einfache Partitionierungen

Dieser Abschnitt besteht aus 2 Teilen, einem abstrakten und einem konkreten. Im ersten Teil geht es um alle Aspekte der Schleifenverhinderung bei DV Routing, für die ein minimales Implementierungskonzept mit IGP-Kopplung angegeben und untersucht wird. Dieses Konzept wird im zweiten Teil prototypisch implementiert, wobei als Zutaten lediglich ein paar open source Router benötigt werden, die mit einigen Verbindungen untereinander und mit Bird als routing suite ausgestattet sind.

IGP-Kopplung mit Schleifenverhinderung - Konzept

Zur Annäherung an den gesuchten Punkt im o.g. Bermudadreick könnte man versuchen, Bellman-Ford in Richtung backbone area bzw. Richtung BGP zu erweitern. Mit so wenig Hilfskontruktionen wie möglich. Im Falle eines link-state Konzepts wie der backbone area erscheint das auf Anhieb nicht gerade vielversprechend, weil die Ausgangsbasis (naive IGP-Kopplung) eben Bellman-Ford realisiert. Also erstmal in Richtung BGP vortasten, das ja als PV Protokoll letzlich zur DV Familie gehört.

Die Lösung des counting-to-infinity Problems wäre eine gute Ergänzung der naiven IGP-Kopplung in Richtung BGP. Das Problem entsteht ja durch die Partitionierung der Routingdomäne. Hätte ein Router Wissen über die gesamte Routingdomäne, könnte er deren Partitionieren bemerken, aber dieses Wissen würde ihn in eine backbone area erheben. Doch dazu fehlt hier noch jede Idee, wie das umgesetzt werden sollte. Aber der Router muss streng genommen nicht die Ursache des Problems erkennen, sondern ihr Symptom: dass nämlich eine Route durch ihn hindurchkreist.

Schleifenverhinderung durch Routenmarkierung

Das kann der Router dadurch erreichen, dass er die Route beim Exportieren so markiert, dass er sie, wenn er sie aus einer seiner anderen IGP-Instanzen wieder zurückbekommt, erkennt - und verwirft. Minimal reicht ein bit in einem bit array, das für jede IGP-Instanz ein bit vorsieht. Bei Erzeugung der Route sind alle bits 0. Das einer betrachteten IGP-Instanz zugeordnete bit setzt ein Router dann auf 1, wenn er die Route aus der betrachteten IGP-Instanz in eine andere IGP-Instanz exportiert.

Sobald man eine solche Mechanik hinkriegt, sind counting-to-infinity und auch etwaige sonstige Routingschleifen offenbar passé. Denn wenn die Route nicht im Kreis läuft, hinterlässt sie in den Routern, von denen sie als Beste akzeptiert wird, FIB Einträge, die als Ganzes betrachtet einen Baum mit dem Quellrouter als Wurzel bilden. (Baum => keine Schleife.) Dieses Verhindern der Schleifenbildung erfolgt konzeptionell durch die Parametrisierung des naiven Bellman-Ford Algorithmus mit folgender feasibility condition: Ein Router nimmt die Route nur an, wenn sie keine Schleifenmarkierung einer IGP-Instanz trägt, an der der Router beteiligt ist.

Wir können also counting-to-infinity durch die Schleifenmarkierung lösen. Aber wie sieht es mit anderen Problemen des DV aus, die in der Babel RFC behandelt werden? Da es sich bei der IGP-Kopplung um parametrisierten und nicht um den naiven Bellman-Ford Algorithmus handelt, sollte man sich das fragen.

route starvation ?

Die route starvation entsteht bei Babel durch ein “Gedächtnis” im Router, der sich die bisher geringste feasibility distance merkt, und mit deren Hilfe nur solche Routen annimmt, die garantiert ohne Schleife als Alternativroute verwendbar wären. Die Grenzrouter einer IGP-Kopplung dagegen merken sich nichts - die Zusatzinformation (das “Schleifenbit”) steckt in der Route, nicht im Router. Außerdem hat jeder Grenzouter im Allgemeinen bereits realisierbare Alternativen: eine Route aus jedem IGP, das einen (schleifenfreien!-) Weg zur Quelle der Route kennt, der nicht über den betrachteten Router führt. Die Kopplungsmechanik in der routing suite erhält ja aus allen IGP-Instanzen, an denen der Router teilnimmt, die jeweils besten Routen in die zentrale Routingtabelle, und wählt daraus die beste aus. Fällt diese weg oder verschlechtert sich ihr Metrikwert, wird eben die bisher zweitbeste aktiviert, die jetzt die beste geworden ist. Diese Änderung wird propagiert, indem die alte Route zurückgezogen und die neue beste Route an die IGP-Instanzen propagiert wird, aus denen sie nicht stammt. Die Nachbarrouter können dann entscheiden, ob auch für sie die beste Route zum Ziel jetzt bzw. weiterhin der betrachtete Router ist. Da die Grenzrouter keine eingebaute Hemmung haben, Veränderungen anzunehmen und ggf. auf Alternativen umzuschalten, ist nicht vorstellbar, wie es bei IGP-Kopplung zu einer route starvation kommen sollte.

Mehrere Quellen für das gleiche Ziel

Wie oben angedeutet verwendet Babel zur schnellen Konvergenz nach Änderungen Alternativrouten, welche aufgrund der feasibility condition als schleifenfrei bekannt sind. Der Preis ist bei Babel die route starvation, deren Lösung wiederum mit dem Preis einer Sequenznummer in der Metrik bezahlt wird: Um aus der starvation wieder rauszukommen, flutet der betroffene Router durch die Routingdomäne einen Hilferuf zum Quellrouter, der diesen veranlasst, die Sequenznummer der Route zu erhöhen, und damit eine neue Route zum Ziel mit besserer Metrik als alle bisherigen zu emittieren.

Um den Hilferuf bei starvation an die richtige Stelle senden zu können, muss die Route eine ID des Quellrouters enthalten. Das führt uns bzw. Babel zum Problem der Mehrfachquellen.

```
2.7. Multiple Routers

   The above discussion assumes that every prefix is originated by a
   single router.  In real networks, however, it is often necessary to
   have a single prefix originated by multiple routers; for example, the
   default route will be originated by all of the edge routers of a
   routing domain.

   Since synchronising sequence numbers between distinct routers is
   problematic, Babel treats routes for the same prefix as distinct
```

```
entities when they are originated by different routers: every route
announcement carries the router-id of its originating router, and
feasibility distances are not maintained per prefix, but per source,
where a source is a pair of a router-id and a prefix.  In effect,
Babel guarantees loop-freedom for the forwarding graph to every
source; since the union of multiple acyclic graphs is not in general
acyclic, Babel does not in general guarantee loop-freedom when a
prefix is originated by multiple routers, but any loops will be
broken in a time at most proportional to the diameter of the loop --
as soon as an update has "gone around" the routing loop.
```

In der IGP-Kopplung mit Schleifenvermeidung enthalten Routen keine Router IDs, sondern außer der (arithmetischen) Metrik nur die Markierungen zur Schleifenvermeidung. Ob eine Route nun von diesem oder jenem Quellrouter emittiert wird, sieht man der Route nicht an. Die Entscheidung eines Routers, ob er den Weg zu diesem oder zu jenem Quellrouter einschlägt, hängt nur von der Entfernung zum Ziel (Metrikwert der Route) ab. Wenn ein Router also von beiden Quellroutern eine Route zum Ziel bekommt, dann entscheidet er sich für die beste von beiden (incl. Berücksichtigung des "Schleifenbits"), und gibt nur diese weiter. So entstehen von jedem Quellrouter aus betrachtet forwarding graphs, die sich nicht überschneiden, weil ein Router "an der Grenze" sich für die eine oder andere Richtung entscheidet, und jede davon funktioniert. Außer, wenn es sich bei einer davon um eine "Gerüchtroute" handelt, die in einer counting-to-infinity Schleife den Zombie gibt. Dies Treiben wird aber dank der "Schleifenbits" gestoppt, d.h. die Gerüchtroute kann keine Schleife laufen. Das Problem der sich überlagernden forwarding graphs entsteht in der (einfachen) Welt der IGP-Kopplung also nicht.

Zur Verdeutlichung folgen wir der Babel RFC noch ein Stückchen im Text:

```
Consider for example the following diagram, where A has selected the
default route through S, and B has selected the one through S':

           1     1     1
::/0 -- S --- A --- B --- S' -- ::/0

Suppose that both default routes fail at the same time; then nothing
prevents A from switching to B, and B simultaneously switching to A.
However, as soon as A has successfully advertised the new route to B,
the route through A will become unfeasible for B.  Conversely, as
soon as B will have advertised the route through A, the route through
B will become unfeasible for A.
```

Bei IGP-Kopplung läuft dieser Vorgang so ab, dass Grenzrouter A die Route über S verliert, und Grenzrouter B die über S'. Nun aktivieren A und B ihre Alternativrouten, die eben über B bzw. A verlaufen. In dieser Situation entsteht eine "micro-loop" zwischen A und B. Aber gleichzeitig ziehen sie die in das gemeinsame IGP jeweils exportierte default-Route, die von S bzw. von S' stammte, zurück (triggered update). Sobald die Information über den retract das IGP zwischen A und B durchquert hat, wird auch die jeweilige Alternativroute bei A und bei B ausgetragen und die "micro-loop" verschwindet. (Das funktioniert sogar ohne Schleifenbits, weil die Topologie des obigen Beispiels linear ist.)

Schleifen durch überlappende Präfixe

Kommen wir also zum letzten Problem des Bellman-Ford Kapitels in RFC 6126, den überlappenden Präfixen. Zitat:

```
2.8. Overlapping Prefixes

   In the above discussion, we have assumed that all prefixes are
   disjoint, as is the case in flat ("mesh") routing.  In practice,
   however, prefixes may overlap: for example, the default route
   overlaps with all of the routes present in the network.

   After a route fails, it is not correct in general to switch to a
   route that subsumes the failed route.  Consider for example the
   following configuration:

              1     1
   ::/0 -- A --- B --- C
```

```
Suppose that node C fails.  If B forwards packets destined to C by
following the default route, a routing loop will form, and persist
until A learns of B's retraction of the direct route to C.  Babel
avoids this pitfall by maintaining an "unreachable" route for a few
minutes after a route is retracted; the time for which such a route
must be maintained should be the worst-case propagation time of the
retraction of the route to C.
```

Hier könnte man schon die Annahme bestreiten, es sei nicht korrekt, einer Route mit allgemeinerem Ziel (::/0) zu folgen, als es das der zurückgezogenen Route war (C). Die gegenteilige Annahme, also dass es korrekt ist, wegfallende speziellere Ziele im Entscheidungsprozess durch allgemeinere Ziele zu ersetzen, implementiert jeder Kernel in Form des most specific forwarding. Man könnte also vermuten, dass die Überlegung des zitierten Abschnitts eher von den Besonderheiten schwieriger Netze wie den WLAN adhoc Meshes motiviert ist, wo Paketverluste das Tagesgeschäft bestimmen. Denn dort kann "the worst-case propagation time of the retraction of the route" ein spürbarer Faktor sein, womit wir uns unverhofft wieder mitten in der langen Fußgängerzone mit den beiden uplinks an den Enden wiederfinden. Und dort überlegen wir: die IGP-Kopplung mag mit der Quantenverschränkung konkurrieren wollen, aber beide schaffen es nicht, (verwertbare:) Information schneller als mit der Lichtgeschwindikeit des Mediums zu übertragen. Das Beste, was man rausholen kann, ist es, die maximale Verbreitungsgeschwindigkeit des Mediums für die Routinginformationen zu erreichen. Die langsameren (niedriger priorisierten) Datenpakete folgen dann wenigstens den besten zZ bekannten Routen. Die im Zitat bemühte routing loop entstünde dann im Idealfall gar nicht, da der retract der spezifischeren Route schneller bei A wäre als die Datenpakete, die via B zu C wollen. Die Datenpakete folgen dann im Beispiel eben der default route. Sinnvollerweise, denn vielleicht kennt ein etwas weiter hinter A liegender Router ja doch noch einen anderen Weg zu C als über den kaputten link zwischen B und C.

Solche holddown Routen sollten zumindest nicht in andere IGPs übertragen werden, weil sie dort - sie sind halt more specific - mglw. noch funktionierende less specific Routen zum Ziel überschreiben.

Ein IGP, das für schwierige Situationen wie WLAN adhoc Meshes die Lichtgeschwindigkeit in diesem zähen Medium anstrebt, muss mit dieser Zähigkeit optimal umgehen. Das schlägt sich in der Babel RFC so nieder:

```
3.7. Sending Updates

   A Babel speaker advertises to its neighbours its set of selected
   routes.  Normally, this is done by sending one or more multicast
   packets containing Update TLVs on all of its connected interfaces;
   however, on link technologies where multicast is significantly more
   expensive than unicast, a node MAY choose to send multiple copies of
   updates in unicast packets when the number of neighbours is small.

   Additionally, in order to ensure that any black-holes are reliably
   cleared in a timely manner, a Babel node sends retractions (updates
   with an infinite metric) for any recently retracted prefixes.
```

Bei WLAN gibt es für Broadcastpakete - im Gegensatz zu unicast - keine Wiederholung bei Nichtbestätigung, also kann man die Kosten für multicast im Vergleich zu unicast hoch ansetzen, und daher unicast nutzen. (Minimale Kosten = Lichtgeschwindigkeit des Mediums.)

Aus der Sicht der IGP-Kopplung ist es die Aufgabe der einzelnen IGPs, den Netzbereich, in dem sie walten, optimal zu durchwalten. Wenn einem IGP mehrfache Wiederholungen von Routinginformation sinnvoll erscheinen, dann soll es so verfahren. Für die Grenzrouter ist nur der Nettoeffekt möglichst schneller Routinginformationsübertragung durch eine solche IGP-Instanz relevant.

Das Problem, das die holddown Routen in einem medienspezifischen IGP lösen, ist auf der Ebene der IGP-Kopplung wenig wirksam. Dank globaler Schleifenvermeidung der Routen hat jedes Paket ein erreichbares Ziel. Bei Routenaggregation allerdings können auch faktisch nicht erreichbare Subnetze in einer aggregierten Route erfasst sein, sodass Datenpakete in dieses faktisch unerreichbare Subnetz fließen wollen. Das ist bei Routenaggregation unvermeidlich und wurde in einem früheren Zitat aus der OSPF Spezifikation bereits gelöst:

```
Before the lookup begins, "discard" routing table entries should
be inserted into the routing table for each of the router's
active area address ranges (see Section 3.5).  (An area range is
considered "active" if the range contains one or more networks
reachable by intra-area paths.) The destination of a "discard"
```

```
    entry is the set of addresses described by its associated active
    area address range, and the path type of each "discard" entry is
    set to "inter-area".[10]

[...]
[10] "Discard" entries are necessary to ensure that route
summarization at area boundaries will not cause packet looping.
```

In dieser Hinsicht sollte (schleifenfreie) IGP-Kopplung wie OSPF areas funktionieren, d.h. Grenzrouter, die lokal aggregierte Routen in andere IGPs geben, müssen für die zugehörigen Präfixe lokal discard Routen installieren, damit eingehende Pakete für diese Präfixe zB mit ICMP unreachable beantwortet werden, was Schleifen der Datenpakete verhindert. Das gilt auch für andere Protokolle mit Aggregation, zB BGP.

Eine anderer Aspekt der Routenaggregation ist ihr Einfluss auf die Metrik. So heißt es in der OSPF Spezifikation:

```
    In order to get better aggregation at area boundaries, area
    address ranges can be employed (see Section C.2 for more
    details).  Each address range is defined as an [address,mask]
    pair.  Many separate networks may then be contained in a single
    address range, just as a subnetted network is composed of many
    separate subnets.  Area border routers then summarize the area
    contents (for distribution to the backbone) by advertising a
    single route for each address range.  The cost of the route is
    the maximum cost to any of the networks falling in the specified
    range.
```

Der letzte Satz macht klar, dass die meisten der aggregierten Subnetze kein optimales Routing erwarten können, sobald die area mehr als einen Grenzrouter hat. Beispiel Fußgängerzone: nehmen wir an, die Router in der Fußgängerzone würden an die dortigen clients per DHCP Adressen aus einem bestimmten Präfix ausgeben. Dann könnten die einzelnen Hostrouten für die clients zu diesem Präfix aggregiert werden, sodass die beiden Uplink-Router (an den Enden der Straße) nur eine Route für den ganzen Präfix in Richtung Backbone melden müssten, was offensichtlich die Menge der dort kursierenden Routinginformationen deutlich reduzieren könnte. Wenn es nur einen client und damit eine host route aus dem betrachteten Präfix gibt, dann wird dieser von beiden Routern so aggregiert, dass die jeweils für den betrachteten Präfix erzeugte Route als arithmetischen Metrikwert den Wert aus der Hostroute zum client übernimmt. Dieser Wert ist für den Router am unteren Ende der Straße niedrig, für den am oberen Ende hoch. Das führt dann tendenziell dazu, dass der Backbone Pakete für den client an den unteren Router schickt - so sollte es sein! Doch wehe, ein weiterer client taucht auf, und zwar am oberen Ende der Straße. Dann hat die vom oberen Router aggregierte Route nämlich weiterhin den hohen Wert des unteren clients, sodass der neue client am oberen Ende in Sachen downstream traffic nix von seiner niedrigen Metrik zum oberen Router hat. Ebenso wird der untere Router nun den hohen Metrikwert zum oberen client in die aggrgierte Route übernehmen. Und schon haben wir wieder das gleiche Problem wie ganz am Anfang unserer Reise durchs wilde Routistan: denn de facto sind die Metriken zwischen Mesh und Backbone nun doch wieder entkoppelt.

Dieser Effekt ist ein prinzipielles Problem von Routenaggregation, und hat erstmal nichts mit IGP-Kopplung zu tun. Sollte diese im FF eingesetzt werden, dann kann sich aber durchaus folgende Situation ergeben: Zwischen Backbone und den WLAN Meshes gibt es noch eine Ebene des Routings ("Supernode Ebene"), die mit den Meshes Metrikkopplung hat, aber zum Backbone hin Routenaggregation verwendet.

Betrachten wir nun zwei Meshes, die jeweils zu einer anderen IGP-Instanz auf der Zwischenebene VPN-Verbindungen haben. Der traffic zwischen diesen Instanzen läuft nun

- über die jeweilige IGP-Instanz der Supernode Ebene,
- dann über den Backbone zur anderen IGP-Instanz der Supernode Ebene,
- und von dort dann (metriktreu) ins andere Mesh.

Die Routenaggregation zwischen Supernode Ebene und Backbone stört hier nicht groß, da die dort mglw. entstehende Suboptimalität angesichts der dort (hoffentlich:) vorhandenen Bandbreiten kaum auffallen wird. Insbesondere wenn man annimmt, dass traffic zwischen verschiedenen Meshes sehr gering ist im Vergleich zum dem, der mit dem Rest des Internet ausgetauscht wird. Wenn die beiden betrachteten Meshes nun aber Nachbarn auf layer 1 sind, zB weil sich einige ihrer Meshrouter gegenseitig gut empfangen, dann könnten sie sinnvollerweise auch direkt miteinander IGP-Kopplung betreiben. Dabei würden sie wohl keine Routenaggregation benutzen, sondern optimales Routing zwischen beiden Bereichen haben wollen. Durch die Routenaggregation zwischen Backbone und "Supernode Ebene"

gibt es in der gesamten Routingdomäne allerdings zwei verschiedene Routen für die Subnetze (incl. Hostrouten) der beiden Meshes, die ursprünglichen Routen und die aggregierten. Dank der "more specific" Methode des Forwardings macht das aber nix: wenn ein Router beide Versionen bekommt, nimmt er die originale (die ist logischerweise more specific als ihr Aggregat.)

Durch die Kopplung "auf der Straße" schließt sich aber ein Kreis, dessen Unterbrechung zu der in der Babel RFC betrachteten Situation führt, dass der Wegfall einer spezifischeren Route zu Routingschleifen führen kann. Aber an unserem Beispiel kann man auch sehen, dass ein holddown für zurückgezogene spezifischere Routen nicht immer das beste Vorgehen zu sein scheint. Wenn im Beispiel der (letzte) Grenzrouter zwischen den Meshes ausfällt oder seine direkte Verbindung zu einem der Meshes verliert, sollte in beiden Meshes so schnell wie möglich auf die aggregierte Route des Nachbarmeshes umgeschaltet werden, damit der traffic zwischen den Meshes noch durchkommt - so wie vor der direkten Kopplung, nämlich über den Backbone.

Schleifenverhinderung versus optimales Routing

Die bisherige Diskussion der möglichen Probleme des DV Routings auf der Ebene der IGP-Kopplung erscheint insgesamt vielversprechend. Durch die Parametrisierung von Bellman-Ford mit den "Schleifenbits" ist aber die Metrik komplex geworden, d.h. die Optimalität der arithmetischen Entfernung wird fraglich. Und tatsächlich kann man sich Beispiele überlegen, in denen die "Schleifenbits" die arithmetische Optimaliät (des naiven Bellman-Ford) stören.

Öffnen wir wieder die Augen und betrachten die Fußgängerzone um uns herum: ab und zu ein FF-Router, mal auf dieser, mal auf jener Seite der Straße. Die Verbindungen zwischen (auch schräg) gegenüberliegenden Routern sind oft besser als die zwischen solchen auf der gleichen Straßenseite. Aus historischen Gründen gehören aber alle Router auf der Westseite der Straße zur selben IGP-Instanz, ebenso wie alle Router auf der Ostseite einem anderen IGP angehören. Also hat man einige Router mit guten Verbindungen zur je gegenüberliegenden Seite zu Grenzroutern konfiguriert, oder hilfsweise von einem allmächtigen Lua-Skript auf dessen Weg zur Weltherrschaft konfigurieren lassen. In dieser zerfrettelten Situation ist es durchaus wahrscheinlich, dass der optimale Weg eines Paketes mehrfach die Straßenseite wechselt - und damit die IGP-Instanz. Eine Route, die von einer IGP-Instanz in die gegenüberliegende wechselt, kann nicht mehr zurück. D.h. die Route - und anschließend in umgekehrter Richtung die zugehörigen Datenpakete - können nur einmal die Straßenseite wechseln, und müssen anschließen die schlechteren Verbindungen längs einer der Seiten entlangkriechen. Die Schleifenvermeidung greift also spürbar in den arithmetischen Teil der Metrik ein, sodass das resultierende Routing nicht mehr im Sinne der ursprünglichen ("naiven") Metrik optimal ist.

Wenn wir also beim IGP-Kopplungs-Marketing weiterhin von "optimalem Routing" jubilieren wollen, dann muss man ehrlicherweise dazusagen, dass diese nur für die Kopplung von arithmetisch konvexen IGP-Instanzen gilt. Damit ist in Analogie zur Geometrie gemeint, dass jeder (arithmetisch) optimale Weg zwischen zwei Routern einer IGP-Instanz vollständig innerhalb der IGP-Instanz selbst verläuft. Diese Anforderung erfüllen die beiden IGP-Instanzen der Fussgängerzone offensichtlich nicht, da ein arithmetisch optimaler Weg (Summe der Kosten/Entfernungen/...) zwischen zwei Routern der Westseite über die IGP-Instanz der Ostseite läuft und umgekehrt.

In der Praxis wird das hoffentlich weniger ein echtes Problem werden, da entweder die Community vor Ort oder das o.g. Lua-Skript (auf seiner nächsten Evolutionsstufe) die Fußgängerzone zu einer gemeinsamen IGP-Instanz zusammenfassen werden.

Exkurs: Toleranz gegen einfache Partitionierungen einer IGP-Instanz

Die Verbindungszahl zwischen den IGP-Instanzen einer IGP-Kopplung ist tendenziell höher, als zwischen Routern innerhalb einer IGP-Instanz. Denn zwei IGP-Instanzen können (und sollen) durch mehrere Grenzrouter verbunden werden, während unabhängige direkte Verbindungen zwischen zwei Routern eher die Ausnahme sind (OK, es kommt vor: 2,4 und 5 GHz gleichzeitig in einem FF-Router (und seinen Nachbarn!)). Daher sollte eine Partitionierung der gesamten Routingdomäne seltener auftreten als die eines IGP. Und so kann man auf die Idee kommen, die Partitionierung einer IGP-Instanz durch den Zusammenhalt des Verbundes aufzufangen.

Damit die (lokale) Partitionierung der IGP-Instanz (global) aufgefangen werden kann, müssen alle Routen der IGP-Kopplung beide Partitionen erreichen. Dies geschieht automatisch, solange die IGP-Kopplung als Ganzes noch zusammenhängend ist. Läuft prima - außer bei den Routen, die aus einer der beiden Partitionen stammen. Denn die sind beim Export aus der Quellpartion heraus mit dem "Schleifenbit" der Quellpartition markiert worden. Und das ist das gleiche Bit wie bei der anderen Partition, denn normalerweise bilden beide ja eine gemeinsame IGP-Instanz. Zwecks Schleifenvermeidung wird die andere Partition die von der Quellpartition stammenden Routen verwerfen. Leider handelt es sich hier um eine Scheinvermeidung, denn es droht gar keine Schleife, die es zu verhindern gälte. Denn wenn die andere Partition die Routen der Quellpartition einlässt, führt das nicht zu Schleifen, sondern dazu, dass die gespaltenen Teile wieder Datenpakete austauschen können - eben außenrum. Aber das ist unter dem Regime "Schleifenbit" nicht machbar, es sei denn ...

Man führt ein einziges weiteres Bit in die Routinginformation ein, nennen wir es re-entry Bit. Und das funktioniert so: Im obigen Beispiel würde ein Grenzrouter der anderen Partition eine schleifenmarkierte Route aus der Quellpartition zwar annehmen, aber dabei das re-entry Bit setzen. Eine so markierte Route wird dann nicht mehr in IGP-Instanzen exportiert, die bereits ihr "Schleifenbit" in der Route gesetzt hatten. Durch dieses einmalige Aussetzen der Schleifenbitauswertung kann die Partitionierung aufgehoben werden. Darüber hinaus kann die Route auch in Netzbereiche "hinter" der anderen Partition gelangen, und dort zum arithmetisch optimalen Routing für die Präfixe aus der Quellpartition beitragen - denn die "Schleifenbits" haben ja noch nicht zugeschlagen.

Trotz der bescheidenen Mechanik der IGP-Kopplung geht sie mit dem re-entry bit über BGP hinaus:-) Denn bei BGP muss ein Router, der die AS Nummer des eigenen AS im Pfad einer eingehenden Route erkennt, diese Route verwerfen: kein re-entry bei BGP. Das entspricht der einfachen Schleifenvermeidung der IGP-Kopplung: jedes "Schleifenbit" steht für eine von der Route bereits besuchte IGP-Instanz, und ein Wiedereintritt der Route in diese Instanz wird verhindert. Das Array der "Schleifenbits" entspricht einer Menge von (verschiedenen) ASNs bei BGP. Das darüber hinausgehende re-entry Bit macht im FF Kontext aber durchaus mehr Sinn als bei den Großen[tm], da gegenseitige Hilfe bei Partitionierung im FF (hoffentlich:) eher gepflegt wird als in einer kommerziellen Konkurrenzsituation. Außerdem dürfte Partitionierung im FF Kontext häufiger auftreten, da eine IGP-Instanz typischerweise kleiner ist und prekärere Verbindungen nutzt, als im AS eines kommerziellen ISP üblich.

Anforderungen an IGPs und IGP-Kopplung

Zum Abschluss des abstrakten Teils dieser "Laborstudie" eine Zusammenstellung der Anforderungen an eine IGP-Kopplung, die uns auf der Reise hierher begegnet sind:

- IGP routet optimal für seinen Einsatzbereich [P,I]
- kann externe Metriken und die Bits zur Schleifenvermeidung transportieren [P,K]
- routing suite kann Schleifenvermeidung durchführen (Setzen und Auswerten der o.g. Bits) [K]
- möglichst zuverlässige triggered updates [P,K]
- discard routes bei Routenaggregation [P,K]
- IGP-Instanzen möglichst konvex [I,K]
- redundante Verbindungen zwischen den IGP-Instanzen, möglichst auch zwischen Nachbarn [I,K]

Dabei ist mit "P" ein Routingprotokoll gemeint, das in einer IGP-Instanz genutzt wird, "I" meint die konkrete Instanz, in der ein bestimmtes Protokoll genutzt wird, und "K" bezieht sich auf die Ebene der IGP-Kopplung als Ganzem, wozu insb. die routing suites auf den Grenzroutern zählen. In der obigen Aufzählung meint [X,Y], dass die Anforderung an die Bereiche "X" und "Y" gestellt wird.

Ehe wir, beladen mit der Anforderungsliste, endgültig zum handgreiflichen Teil der Reise hinabsteigen, soll noch einmal der Blick zurück geworfen werden, um eine zusammenfassende Einordnung der IGP-Kopplung zu versuchen.

Ausgehend vom Problem der entkoppelten Metriken (der Fußgängerzone) sind wir durchs lichte Tal des link-state Routing gewandelt, konnten aber die Höhen der backbone area mit unserer leichten Ausrüstung nicht erklimmen. So zogen wir weiter in die verworrenen Wälder des distance-vector Routing, trugen aber die Erinnerung an das angenehm optimale LS Tal im Herzen. In den Wäldern nun erwies sich unsere leichte Ausrüstung als gar nicht übel, da sie aus den Wäldern stammt. Da wir jetzt wussten, wie wir reisen wollten, konnten wir fast so angenehm durch die DV Wälder reisen, wie durch das LS Tal. Unterwegs haben wir aus den Begegnungen mit OSPF, Babel und BGP viel lernen können. Nun kehren wir mit diesem Wissen in die Fußgängerzone zurück, um auch in ihr eine angenehme Metrik zu schaffen.

IGP-Kopplung mit Schleifenverhinderung - Implementierung eines Prototypen

Kommen wir zuerst zu den grundlegendsten Anforderungen an die Bestandteile einer IGP-Kopplung, denn sie bestimmen am meisten die Auswahl der in Frage kommenden Mittel für eine Implementierung.

Konsequenzen aus den Anforderungen

- Dass ein IGP optimal routet, sollte klar sein, lässt aber das in allen routing suites vorhandene R.I.P. ausscheiden. Und das Protokoll sollte zur konkreten IGP-Instanz passen, also zB kein OSPF über WLAN adhoc Meshes. ((Das machen wir trotzdem, aber das ist ein Detail in den untigen Niederungen.))

- Den Transport externer (arithmetischer) Metriken und Zusatzinformationen kennen von den üblichen Verdächtigen nur BGP und OSPF. Wir wollten hinsichtlich der Mittel aber unterhalb von BGP bleiben, bleibt also aktuell nur OSPF als Kandidat für das "IGP" in "IGP-Kopplung". (Natürlich könnte man auch BGP-Instanzen integrieren, aber das muss man sich nicht antun, wenn man auch OSPF haben kann.) Wir hoffen derweil auf die erfolgreiche Integration von Babel in Bird, dann wird's an dieser Stelle nochmal spannend.
- Die "Intelligenz" einer IGP-Kopplung sitzt offenbar in der routing suite selbst, genauer gesagt in ihrer Möglichkeit, Routen zu filtern und zu manipulieren. Wenn sich die IGP-Kopplung überhaupt mit aktuell verfügbaren Mitteln umsetzen lässt, dann sollte man es zuerst mit der in dieser Hinsicht flexibelsten routing suite versuchen, also mit Bird.

Jetzt müssen wir nur noch prüfen, ob Bird + OSPF als dream team in den Ring steigen sollten. Gehen wir mit ihnen also die restlichen Anforderungen durch:

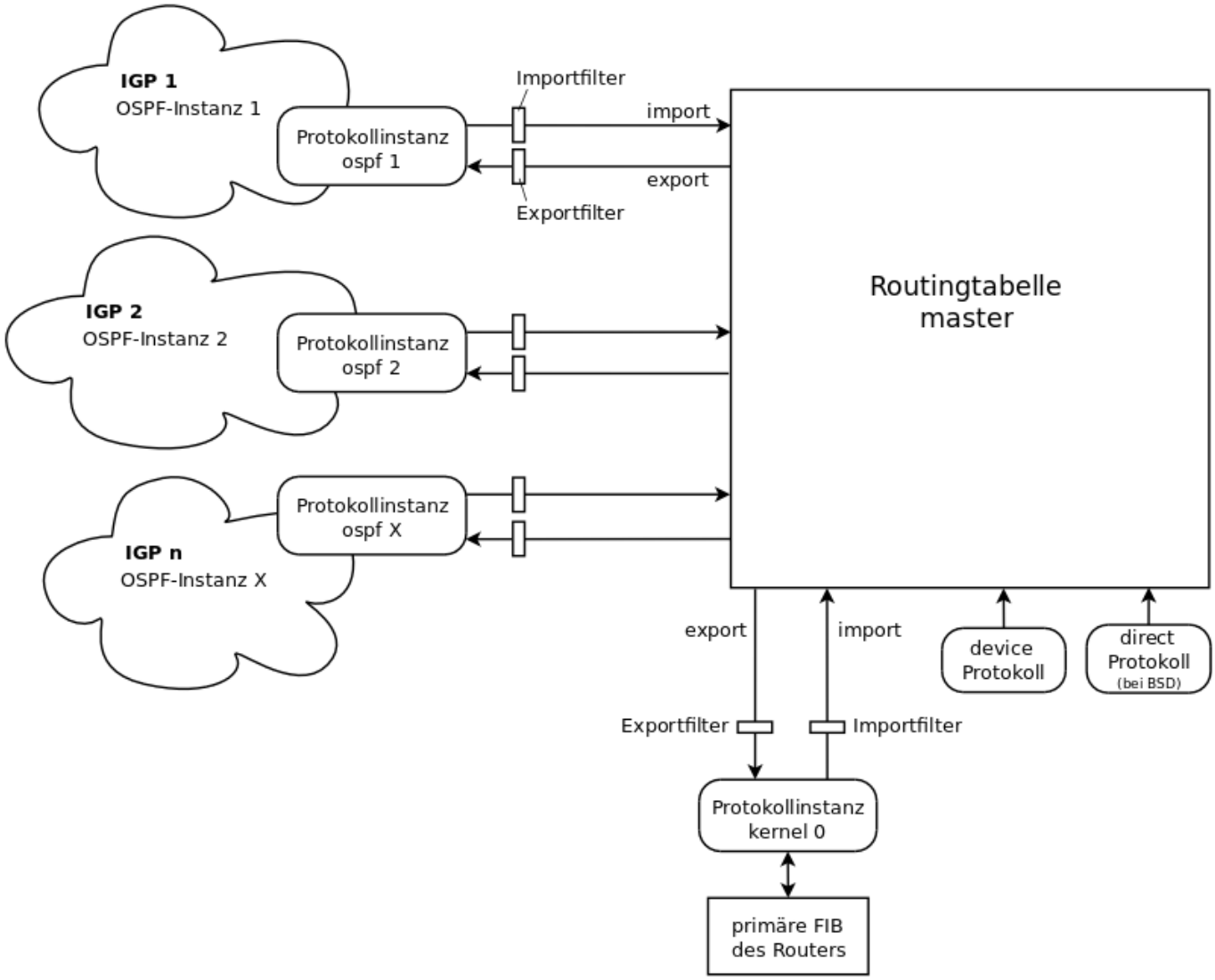
- OSPF kommt aus dem LS-Tal und wird für seine schnelle Konvergenz geschätzt, sodass triggered updates in den OSPF-Instanzen als gelöstes Problem angesehen werden können. Und Bird kümmert sich intern um das Weiterleiten von Routen und ihren Änderungen, was bei Bird (klein, leicht, schnell) recht flott gehen sollte.
- Routenaggregation werden wir bei unserem kleinen Prototypen erstmal nicht brauchen, aber OSPF macht das, wenn es an area Grenzen aggregiert. Die Bird-Mechanik selbst macht keine Routenaggregation, sodass auch von dieser Seite keine discard routes gesetzt werden müssten.
- Die IGP-Instanzen des Prototyps bestehen nur aus je einem link zwischen Grenzroutern, sind also trivialerweise konvex.
- Aufgrund des quantitativ prekären Versuchsaufbaus entfällt die Anforderung redundanter Verbindungen zwischen IGP-Instanzen. Aber prinzipiell könnte es auch mehr als einen Grenzrouter zwischen benachbarten IGPs geben, womit weder Bird noch OSPF ein Problem hätten. Bei dieser Anforderung handelt es sich eher um ein Zitat aus der Bedienungsanleitung der IGP-Kopplung, die den Netzbetreibern nahelegt, die Stabilität des globalen Zusammenhangs durch Verbindungen zwischen den IGP-Instanzen zu verbessern.

Aus der Verwendung von OSPF innerhalb einer IGP-Kopplung folgt bereits deren maximale Größe: 31. Denn OSPF kann nur 32 Bit an Zusatzinformation in einer Route transportieren, sodass wir zwecks Schleifenvermeidung 32 - 1 = 31 IGP-Instanzen unterscheiden können, die "- 1" ist das re-entry Bit.

Der Weg durch die verworrenen Wälder des DV hat Vertrauen in die leichte Ausrüstung (und uns selbst;) geschaffen, und so wagen wir uns an den ungewissen Abstieg.

inside Bird: Routingtabellen, Protokollinstanzen und Filter

Wer hier weiterlesen möchte, aber noch keine Erfahrung mit Bird hat, sollte zunächst den Abschnitt About routing tables in der Bird Doku verstehen. Die Konfiguration der einzelnen Protokolle wird auf der Seite Protocols beschrieben; wir benutzen davon kernel, device, direct, ospf (und für einen workaround auch pipe und bgp, s.u.). Eine Protokollinstanz wird mit einer Routingtabelle verbunden, der default dafür ist "master". Über diese Verbindung werden Routen zwischen Tabelle und Protokoll ausgetauscht, wobei die beiden Richtungen als "import" bzw. "export" bezeichnet werden. Die Bezeichnungen beschreiben die Richtung von der Tabelle aus gesehen, d.h. "import"iert werden Routen vom Protokoll in die Tabelle, "export"iert werden Routen von der Tabelle ins Protokoll (und über dieses dann an andere Router). Auf dem Weg zwischen Tabelle und Protokoll können die Routen von Filtern geblockt oder manipuliert werden. Je nach Richtung spricht man von Importfiltern bzw. Exportfiltern. Zur Formulierung der Filter gibt es in Bird eine einfache Filtersprache ("Routing-Assembler":-), deren Doku man sich ebenfalls bookmarken sollte.



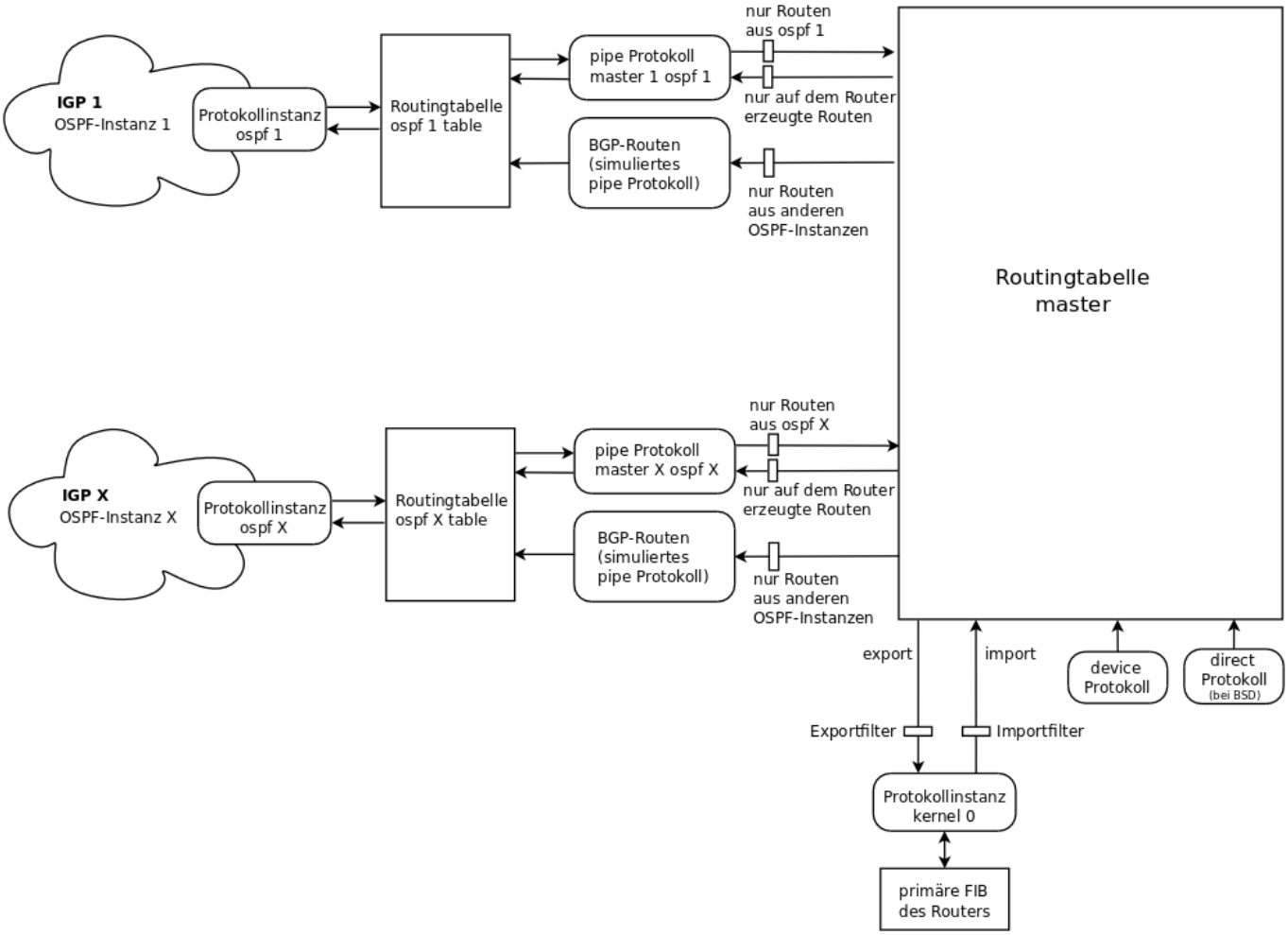
Man nehme:

- Als primäre Routingtabelle benutzen wir im Folgenden die vordefinierte mit dem Namen “master”.
- Diese wird über die Protokollinstanz “kernel0” mit der primären FIB im Kernel des Routers verbunden.
- Außerdem wird die Routingtabelle mit einem device Protokoll verbunden, um Informationen über interfaces für Protokolle verfügbar zu machen, zB link down/up. um Routen für die direkt an den Router angeschlossenen interfaces zu importieren. (Diese brauchen nicht via “kernel0” an die FIB exportiert zu werden, denn der Kernel kümmert sich schon selbst um interface Routen.)
- Da der Prototyp auf Routern mit einem BSD als OS läuft, brauchen wir noch das direct Protokoll, dass aus den IP-Konfigurationen lokaler interfaces Routen für deren Präfixe erzeugt.
- Für jede OSPF-Instanz, an der der Router teilnimmt, wird entsprechend in Bird eine OSPF-Instanz konfiguriert: “ospf1” . . . “ospf5”.
- Für jede OSPF-Instanz werden Filter definiert. Hier wird zunächst geregelt, welche Routen überhaupt zulässig sind. Darüber hinaus können hier Metrikumrechnungen und die für IGP-Kopplung wichtigen Schleifenverhinderungsmechanismen implementiert werden.

Leider reicht dieser Grundaufbau zZ nicht aus, wie im nächsten Abschnitt zu lesen sein wird. Wenn sich Bird so patchen lässt, dass er OSPF<->OSPF Redistribution zulässt, dann können die folgenden Umständlichkeiten wegfallen. Was wir jetzt machen werden, ist allerdings nicht nur umständlich, sondern auch lehrreich für die Möglichkeiten, die Bird bietet.

- Für jede OSPF-Instanz wird eine separate Routingtabelle definiert: “ospf1table” . . . “ospf5table”.
- Jede OSPF-Instanz wird mit ihrer jeweiligen Routingtablle verbunden, statt mit der “master” Tabelle.

- Diese Routingtabellen werden jeweils über ein pipe Protokoll mit der “master” Tabelle verbunden: “master1ospf1”, . . . , “master5ospf5”.
- Der Importfilter eines solchen pipe Protokolls lässt nur die Routen in die “master” Tabelle, die aus der zugehörigen OSPF-Instanz stammen.
- Der Exportfilter eines solchen pipe Protokolls lässt nur die Routen aus der “master” Tabelle in die instanzspezifische Tabelle, die lokal auf dem Router erzeugt wurden, aber *nicht* solche, die aus einer anderen OSPF-Instanz stammen.
- Fehlen also noch solche OSPF-Routen, die von einer OSPF-Instanz in die betrachtete weitergereicht werden sollen, also das Kerngeschäft einer IGP-Kopplung. Zwecks workaround (s.u.) simulieren wir ein pipe Protokoll, welches die OSPF-Routen der “master” Tabelle beim Export in die instanzspezifische Tabelle in BGP-Routen verwandelt, die im Wesentlichen die gleichen Routinginformationen tragen wie die OSPF-Routen. Diese können dann aus der instanzspezifischen Tabelle in die zugehörige OSPF-Instanz exportiert werden.



Es ist auch für die Übersicht über das Routinggeschehen in einer IGP-Kopplung hilfreich, in einem Router alle Routen in einer Tabelle beisammen zu haben, die zu einer bestimmten IGP-Instanz gehören - importierte und exportierte.

Der nächste Abschnitt beschäftigt sich mit dem tieferen Grund für den workaround sowie seinem Implementierungskonzept. Wer “nur” an der eigentlichen IGP-Kopplung interessiert ist, kann den folgenden Abschnitt über den workaround auslassen.

Warum nicht gleich alle IGP-Instanzen mit der zentralen Routingtabelle verknüpfen?

Um zwei IGP-Instanzen per IGP-Kopplung miteinander zu verbinden, müssen Routinginformationen von einer in die andere gelangen. Das sind im Falle der IGP-Kopplung mit Schleifenverhinderung die (arithmetische) Metrik und die “Schleifenbits”. Würde eine Instanz BGP sprechen, die andere OSPF, so könnten im Exportfilter aus den Attributen der BGP-Route diejenigen für die OSPF-Route berechnet und gesetzt werden. ZB könnte der Wert des BGP-Attributs “bgp_med” in das OSPF-Attribut “ospf_metric1” kopiert werden.

Bird's Problem mit der OSPF<->OSPF Redistribution

Doch beim Versuch, eine OSPF-Route in eine (andere) OSPF-Instanz zu exportieren, löscht Bird (1.5.0) die relevanten Attribute der Route, nämlich die Metrikwerte sowie den "ospf_tag", in den wir die "Schleifenbits" kodieren wollen. Dafür mag es Gründe geben, aber vmtl. funktioniert hier BGP<->OSPF schon einfach deshalb, weil das in der Praxis ein normaler Anwendungsfall von Bird ist. Aber ebenso vmtl. hat noch niemand ernsthaft versucht, zwei verschiedene OSPF-Instanzen zu koppeln, da es in der Praxis nur eine OSPF-Instanz in einem AS gibt. Das ist durchaus verständlich, wie wir weiter oben gesehen hatten, denn OSPF-areas haben ja Modell gestanden für die IGP-Kopplung, für die allerdings eine backbone area noch utopisch ist. Weil also eine OSPF-Instanz aus beliebig vielen areas bestehen kann, die schleifenfrei durch die backbone area zusammengehalten werden, braucht man in einer Situation, wo OSPF (neben iBGP) das einzige IGP ist, normalerweise keine IGP-Kopplung. (Das Projekt "Prototyp einer IGP-Kopplung" fällt offenbar nicht in die Kategorie "normal";-)

Bevor man eine IGP-Kopplung mit mehr als einer OSPF-Instanz produktiv betreiben will, sollte der untige workaround durch eine Modifikation von Bird überflüssig gemacht werden. Wenn man so einen patch erarbeitet, kann es hilfreich sein, bereits einen (dank workaround) funktionsfähigen Prototypen zu haben, gegen den man seinen gepatchten Bird (ohne workaround) testen kann.

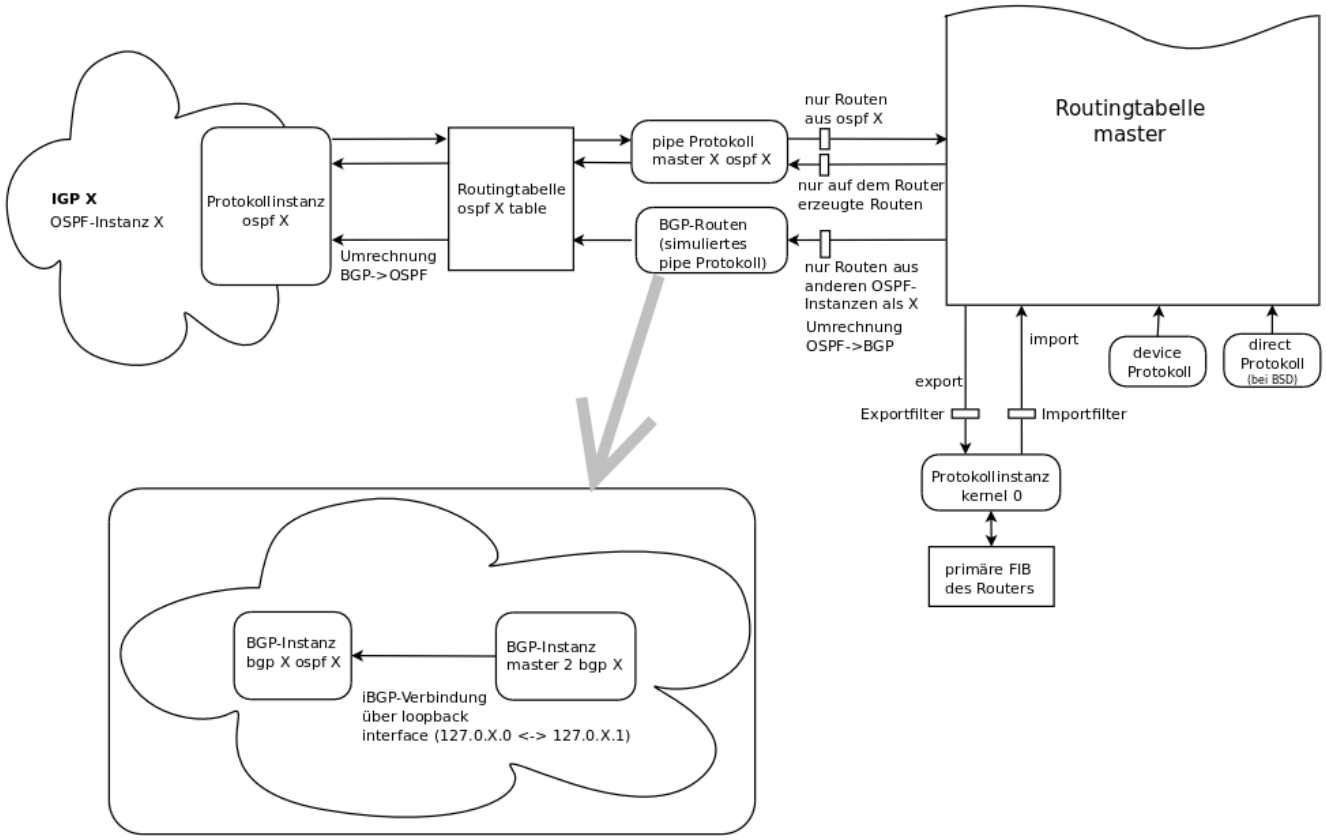
Koste es, was es wolle: ein workaround für das OSPF<->OSPF Problem

Wir beißen also in den sauren Apfel und exportieren die OSPF-Routen "irgendwie" von einer Instanz in die andere. Wir wählen "irgendwie" = BGP, da unter den Bird-Protokollen außer OSPF nur BGP genug Routinginformation für unsere Zwecke transportieren kann, und BGP<->OSPF in der Praxis üblich ist und daher von Bird unterstützt wird.

Man nehme je OSPF-Instanz X:

- Eine BGP-Instanz, die mit der "master" Tabelle verbunden wird: "master2bgpX".
- Eine weitere BGP-Instanz, die mit der (OSPF-)instanzspezifischen Tabelle verbunden wird: "bgpXospfX".
- Ein loopback interface des Routers, über das die beiden genannten BGP-Instanzen eine TCP Verbindung aufbauen können.
- Die beiderseitige Konfiguration einer iBGP-Verbindung zwischen den beiden BGP-Instanzen "master2bgpX" und "bgpXospfX".
- Einen Exportfilter für das Protokoll "master2bgpX", der nur OSPF-Routen von der "master" Tabelle exportiert.
  - Dieser Filter blockiert non-OSPF-Routen wie zB device Routen, da diese schon vom pipe Protokoll "masterXospfX" übertragen werden.
  - In diesem Filter müssen die für die IGP-Kopplung relevanten Attribute von der ursprünglichen OSPF-Route in Attribute einer BGP-Route umgerechnet werden.
- Einen Exportfilter für die OSPF-Instanz "ospfX", der die Attribute aus den BGP-Routen, die von "bgpXospfX" in die instanzspezifische Tabelle "ospfXtable" importiert wurde, wieder zurück in OSPF-Attribute abbildet.

Insgesamt sieht die Konstruktion so aus:



Dabei ist noch ein Schleifenproblem zu behandeln: Routen, die von der OSPF-Instanz ospfX in die “master” Tabelle importiert wurden (per pipe Protokoll “masterXospfX”), dürfen nicht per iBGP zurück in die instanzspezifische Tabelle ospfXtable importiert werden. Denn von dort könnten sie wiederum Richtung “master” oder “ospfX” wandern, wo sie ursprünglich herkamen. Normalerweise kümmert sich Bird darum, dass eine Route zwischen einer Routingtabelle und einer Protokollinstanz nicht sowohl importiert als auch exportiert wird. Durch unseren workaround existiert die Route aber einmal als (originale) OSPF-Route, später dank iBGP aber auch als BGP-Route in der “ospfXtable”. Daher wird in die Exportfilter des pipe Protokolls “masterXospfX” und des BGP-Protokolls “master2bgpX” eingebaut, dass sie solche “Rundläufer” blockieren.

Damit endet der Exkurs zum workaround um das OSPF<->OSPF Problem von Bird zu umgehen. Wenn sich genug aktives Interesse an der IGP-Kopplung mit Bird entwickelt, sollte sich auch eine Flexibilisierung in Bird selbst einbringen lassen (patch). Denn dadurch würde sich die Konfiguration für eine IGP-Kopplung massiv vereinfachen, und auch der Laufzeitaufwand würde reduziert: wesentlich weniger Routingtabellen und Protokollinstanzen als mit dem workaround.

Übertragung der Metriken

Zentrale Idee der IGP-Kopplung ist die Verwendung einer globalen Metrik, wodurch optimales Routing erreicht werden kann, ohne dass Topologieinformationen einer beteiligten IGP-Instanz außerhalb derselben kursieren. Eine Umrechnung der Metriken muss beim Import von einer IGP-Instanz in Richtung “master” Tabelle erfolgen, damit die Routen in der “master” Tabelle vergleichbar sind, d.h. ihr arithmetischer Metrikwert ist in der globalen Metrik ausgedrückt.

Wir gehen in diesem Prototyp, der durch den workaround schon aufgebläht genug ist, aber davon aus, dass die IGP-Instanzen Kosten sowieso in der globalen Metrik ausdrücken. Das erspart auch Kopfrechenkrebs bei der Analyse von Routingtabellen von IGP-Instanzen mit nicht-globaler Metrik.

Was zu tun bleibt, ist also die 1:1 Übertragung der arithmetischen Metrikwerte.

- Das “ospfX” Protokoll exportiert eine OSPF-Route mit “ospf_metric1” in die “ospfXtable”, von dort geht sie unverändert durch das pipe Protokoll “masterXospfX” in die “master” Tabelle. Fehlt noch der umgekehrte Weg:
- “master” -> “ospfX” führt durch “master2bgpX”, wobei - wie oben zum workaround schon erwähnt - die “ospf_metric1” aus der “master” Tabelle in das “bgp_med” Attribut der nach BGP exportierten Route kopiert wird. Auf der Empfängerseite des iBGP erscheint die die BGP-Route unverändert. Sie wird dann beim Export von der “ospfXtable” in die OSPF-Instanz “ospfX” zur OSPF-Route, wobei der Exportfilter das “bgp_med” Attribut in die “ospf_metric1” kopiert.

Exkurs: cold potato und hot potato

(Dieser Exkurs sollte nur bei fortgeschrittenem Interesse an den Möglichkeiten von IGP-Kopplung gelesen werden.)

Es soll nicht verschwiegen werden, dass OSPF zwei Arten von externen Routen kennt: die mit interner, und die mit externer Metrik. Die interne Metrik (Routentyp "E1") ist diejenige, welche die Pfadkosten innerhalb der OSPF-Instanz ausdrückt. Die externe Metrik (Routentyp "E2") drückt die Pfadkosten aus, die vom Grenzrouter der Instanz aus bis zum eigentlichen Ziel anfallen. Wenn eine OSPF-Instanz eine externe Route (von einem anderen Protokoll) erhält, und sie als "E2"-Route in diese OSPF-Instanz einbringt, wird der arithmetische Metrikwert der externen Route als Attribut "ospf_metric2" gesetzt. (Bei Bird wird eine Route beim Export in eine OSPF-Protokollinstanz eben dadurch zur "E2"-Route, dass der Filter für das Attribut "ospf_metric2" einen Wert setzt.) Die Bedeutung von Routen des Typs "E2" im OSPF Protokoll liegt im Entscheidungsverfahren der OSPF Router:

- Bei Zielen, die nur aus "E2"-Routen bekannt sind, wähle denjenigen Grenzrouter als internes Ziel der Route, der die "E2"-Route mit der kleinsten "ospf_metric2" eingebracht hat. Sollten dies mehrere Grenzrouter tun, wähle den mit den geringsten internen Pfadkosten (welche sowohl bei "E1" als auch bei "E2"-Routen in der "ospf_metric1" akkumuliert werden).
- Ziele von "E1"-Routen sind (normalerweise) direkt an den Grenzrouter angeschlossene Netze. Daher ziehe Routen vom Typ "E1" immer denjenigen vom Typ "E2" vor. D.h. jede (noch so große) "ospf_metric1" einer "E1"-Route ist besser als jede (noch so kleine) "ospf_metric2" einer "E2"-Route.

Sinn der Übung ist es, Ziele außerhalb der eigenen OSPF-Instanz optimal zu routen - aber "optimal" im Sinne des "Außerhalb". Dadurch kann man ein Routing erreichen, welches zB in der (früher zitierten) RFC zum Thema BGP-MED als "cold potato" Routing bezeichnet wird. Betrachten wir zwei IGP-Instanzen, die durch mehrere Grenzrouter miteinander verbunden sind. Die erste verfügt über reichlich Übertragungskapazität, die zweite ist dahingehend eher prekär aufgestellt. Wenn nun die erste (üppige) IGP-Instanz Routen von zweiten (kargen) als "E2"-Routen übernimmt, dann wird optimal bezüglich der Metrik der kargen Instanz geroutet: in der üppigen Instanz wird (gemäß obigem Entscheidungsverfahren) jeweils der Grenzrouter ausgewählt, von dem aus der kürzeste Weg innerhalb der kargen Instanz zum Ziel führt. (Im FF-Kontext wäre ein WLAN adhoc Mesh ("Fußgängerzone") ein üblicher Verdächtiger für den kargen Part in dieser Konstruktion.)

Durch Anwendung von cold potato weicht man vom global optimalen Routing ab - weil man es so will, damit die Starken den Schwachen helfen können. Um dies im Rahmen einer IGP-Kopplung leisten zu können, muss der Exportfilter, über den eine Route in eine OSPF-Protokollinstanz gelangt, wissen, ob er die Route als "E1"- oder als "E2"-Route ins OSPF einbringen soll. Ob eine OSPF-Instanz cold potato Routing macht, muss also in der instanzspezifischen Konfiguration auf den Grenzroutern dieser Instanz festgelegt sein. Wenn die Route nach einem Transit durch die üppige OSPF-Instanz dann in weitere IGP-Instanzen übergeht, so muss dort beim Import folgendes geschehen: die Transitroute (vom Typ "E2") hat auf dem (exit-)Grenzrouter zwei arithmetische Metrikwerte: "ospf_metric2" und "ospf_metric1". Wenn diese OSPF-"E2"-Route dann auf dem Weg in die nächste IGP-Instanz zur BGP-Route wird (wg. iBGP im workaround), dann bilde die "bgp_med" der BGP-Route als Summe von "ospf_metric2" und "ospf_metric1" der ursprünglichen OSPF-"E2"-Route. Dadurch werden die tatsächlichen Pfadkosten vom Grenzrouter bis zum Ziel (in der kargen IGP-Instanz) an die nächste IGP-Instanz weitergegeben, zwecks Transitivität der Metrik bei Transit.

Wie zu befürchten, gibt es nicht nur cold potato, sondern auch hot potato. Das ist meist das, was man auch in der Fußgängerzone haben will: die Pakete sollen auf dem kürzesten Weg durch die Luft ins Kabel. Das lässt sich dadurch erreichen, dass beim Einbringen einer Route aus der benachbarten "Kabel-Instanz" der IGP-Kopplung, an der das mesh hängt, deren arithmetischer Metrikwert auf 0 (Null) gesetzt wird. Gut für die Fussgängerzone, doch schlecht für den Transit: die ursprüngliche (externe) Metrik geht dabei verloren, sodass andere IGP-Instanzen, mit der die Fußgängerzone gemeinsame Grenzrouter hat, die Route nicht übernehmen dürfen, da ihr arithmetischer Metrikwert intransitiv zustandekam, also für Transit unbrauchbar geworden ist. Wenn man dennoch hot potato für die Fußgängerzone haben will, kann man es aber auch im Prototypen konfigurieren - instanzspezifisch wie bei cold potato oben. Ob das eine Verbesserung gegenüber dem default (= optimales Routing = best potato:) bringt, wird von den arithmetischen Metrikwerten der Wege innerhalb des Mesh abhängen. Sind diese deutlich größer als die auf der Kabelseite verwendeten Werte, so sollte auch dadurch schon der hot potato Effekt erzielt werden können. Gut zu wissen, wenn man doch mal Transit durch die Fußgängerzone schicken möchte (zB bei einer Partitionierung auf der Kabelseite).

Fazit des Exkurses: Die von den Großen[tm] inspirierte Verwendung des BGP-MED Attributs lässt sich auf IGP-Kopplung übertragen, um auch cold und hot potato Routing zu ermöglichen. Das ist für die IGP-Kopplung kein primäres Problem, da sie normalerweise mit global optimalem Routing arbeitet. Aber für das nachbarschaftliche Feintuning des Routings sollte man diese Möglichkeiten in Erinnerung behalten und an "echten Fußgängerzonen" experimentell überprüfen.

Schleifenverhinderung: Bitarithmetik mit dem "Routing-Assembler"

Die Implementierung der Schleifenverhinderung durch "Schleifenbits" geschieht logisch gesehen an zwei Stellen: beim Setzen und beim Auswerten der Bits.

- Das Setzen eines "Schleifenbits" für die IGP-Instanz X implementieren wir beim Export einer Route mit Ursprung "ospfX" aus der "master" Tabelle in Richtung einer bestimmten IGP-Instanz Y (Y != X). Der entsprechende Filtercode für das Setzen von "Schleifenbit X" findet sich also im Exportfilter der Protokollinstanz "master2bgpY".
- Das Auswerten von "Schleifenbit" X erfolgt zunächst beim Import von "ospfY" in Richtung der "master" Tabelle. Über alle IGP-Instanzen der IGP-Kopplung, an denen der Grenzrouter teilnimmt, wird geprüft, ob eine von ihnen die IGP-Instanz X ist (deren "Schleifenbit" ja in der betrachteten Route gesetzt ist).
- Wenn das "Schleifenbit" X gesetzt ist, gibt es zwei Möglichkeiten:

  a) Das re-entry Bit in der Route ist nicht gesetzt. Dann wird die Route in Richtung "master" durchgelassen, aber das re-entry Bit wird dabei gesetzt.

  b) Das re-entry Bit in der Route ist gesetzt. Dann wird die Route blockiert. Der zughörige Filtercode befindet sich im Importfilter der OSPF-Protokollinstanz "ospfY" (Y != X).

Dadurch können, wie weiter oben beschrieben, einfache Partitionierungen umgangen werden, aber dennoch gerät die Route nicht in einen Kreislauf (und daher kein counting-to-infinity).

Kommen wir zum Kleingedruckten der "Schleifenbits", ihrer Kodierung im Attribut "ospf_tag" einer OSPF-Route. Die einfachste Methode ist die Zuordnung von IGP-Instanz N zum Bit an der Stelle N, wobei als eindeutige ID für eine Instanz eine natürliche Zahl N < 32 verwendet wird. Wir modifizieren das etwas:

- Die untersten (niedrigwertigsten) 5 Bits werden zur Kodierung der Quell-IGP-Instanz dieser Route verwendet (source ID). Diese Bits 0...4 nennen wir "Quellbits".
- In den Bits 5...31 kodiert der Wert 1 an der Stelle k (5 <= k <= 31) den Durchlauf der Route durch IGP-Instanz N = k-4 (1 <= N <= 27). Handgreiflich kann man sich das so vorstellen, dass ein Läufer beim Weitergeben des Staffelstabes "Route" seine individuelle Kerbe in den Staffelstab ritzt, anhand derer er später erkennen kann, dass er diesen Staffelstab schonmal in der Hand hatte. Diese Bits 5...31 nennen wir "Kerbenbits".
- Upps, wir haben das re-entry Bit vergessen. Das einfachste wäre, dafür zB das Bit 5 oder 31 zu verwenden. Dann hat man zwar nur noch 26 "Kerbenbits", aber die reichen ja auch erstmal. Andererseits haben wir uns aus pragmatischen Gründen (s.u.) schon die "Quellbits" gegönnt, und die machen es genau betrachtet überflüssig, das "Kerbenbit" der Quell-IGP-Instanz zu benutzen, da man schon den "Quellbits" entnehmen kann, dass die Route bereits in der Quell-IGP-Instanz war. Das re-entry Bit einer Route wird also im "Kerbenbit" der Quell-IGP-Instanz kodiert.

Die Komplikation durch die "Quellbits" hat rein pragmatische Gründe: die Information, woher eine Route letztlich stammt, ist für Analysen und insb. Fehlerlokalisierung immer, also auch im Betrieb einer IGP-Kopplung, äußerst hilfreich. Und im FF-Kontext mglw. unverzichtbar, da es dort eher vorkommt als im kommerziellen Kontext, dass ein zusammenhängendes Netz mit gemeinsamem Routing (die IGP-Kopplung) von unterschiedlichen Organisationen oder Gruppen (wie zB FF-Communities) betrieben werden.

Zur Umsetzung des obigen "Kleingedruckten" müssen wir mit dem minimalistischem Bird Filtercode ("Routing-Assembler") Bitarithmetik durch Integerarithmetik emulieren. Bird hat keine Bitarithmetik - im Gegensatz zu einem "echten" Assembler:-) Die Kernoperationen unserer Emulation sind:

2 ^ n, n div (2 ^ m), n mod (2 ^ m)

- 2 ^ n können wir in Bird weder rekursiv noch iterativ berechnen, da diese Konzepte in Bird (absichtlich!) nicht vorhanden sind. Da wir aber nur mit 32 Bit langen Größen zu tun haben ("ospf_tag"), können wir uns mit einer Fallunterscheidung behelfen, die für gegebenes n zwischen 0 und 31 den (konstanten:) Wert 2 ^ n zurückgibt. In Bird also ein case switch mit den Alternativen 0...31.
- n div (2 ^ m) entspricht einem Rechtsshift um m Bits. Die Integerdivision von Bird, der Operator "/", tut genau das, und 2 ^ m können wir ja auch schon.
- n mod (2 ^ m) entspricht einem Weglassen aller Bits außer den untersten m Bits. Wir verwenden die Gleichung n mod m = n - ((n div m) * m)

Das initiale Setzen der “Quellbits” der Route (beim ersten Verlassen der Quell-IGP-Instanz) kann durch einfache (Integer-)Wertzuweisung der ID der Quellinstanz an den “ospf_tag” erfolgen, weil die source ID in die niedrigwertigsten Bits kodiert wird. Durch diese Zuweisung werden auch alle “Kerbenbits” auf 0 (Null) gesetzt. Das Auslesen der “Quellbits” erfolgt durch Weglassen der Bits 5...31, also durch die Operation n mod (2 ^ 5) = n mod 32.

Das Setzen des “Kerbenbits” Nr. N (1 <= N <= 27) erfolgt durch Addition von 2 ^ (N + 4). Dabei ist sicherzustellen, dass dieses “Kerbenbit” vorher noch nicht auf 1 gesetzt war, denn sonst würde es durch diese Addition wieder auf 0 zurückgesetzt, und der entstehende Übertrag der Addition würde auch noch das oder die nächsthöheren “Kerbenbits” ungewollt invertieren, d.h. die Mechanik der Schleifenverhinderung sabotieren. Das Auslesen des “Kerbenbits” Nr. N geschieht mit Hilfe der Gleichung (n div (2 ^ (N + 4))) mod 2 = 1, also nach Rechtsshift um N + 4 Bits prüfen, ob das niedrigstwertige Bit gesetzt ist.

Damit haben wir (endlich;) alles beisammen, um die konkrete Konfiguration einer IGP-Kopplung anzugehen.

Struktur der resultierenden Bird Konfiguration

Die Konfiguration können wir ganz abstrakt in vier Ebenen gliedern:

- Das, was jede IGP-Kopplung braucht (zB Funktionen für die Bitarithmetik).
- Das, was für die konkrete IGP-Kopplung spezifisch ist.
- Das, was für jede IGP-Instanz spezifisch ist.
- Das, was für jeden Grenzrouter spezifisch ist.

Wir behalten das im Hinterkopf, wenn wir nun die Struktur der Konfiguration durchgehen:

- Die zentrale Konfigurationsdatei, bird.conf, beginnt mit den Definitionen für die Bitarithmetik, da diese im nächsten Abschnitt bereits benötigt werden.
- Nun werden wesentliche Definitionen des gerade zu konfigurierenden Grenzrouters inkludiert (aus bird.local.conf). Vor allem die Router ID, aber auch die Funktionen fn_is_local_ospf_instance (int -> bool) und fn_any_local_instancebit_is_set_in_ospf_tag (() -> bool). Der Grenzrouter muss eben wissen, zu welchen IGP-Instanzen er gehört, denn nur mit diesem Wissen kann die Schleifenverhinderung Schleifen verhindern. Da die genannten Funktionen auf die Bitarithmetik zurückgreifen, mussten diese im vorigen Schritt definiert werden. Weiterhin gibt es die routerspezifische Konstante Unrouted_local, in der solche Präfixe aufgelistet sind, die der Router zwar lokal kennt, die aber nicht am Routing (irgend)einer IGP-Instanz teilnehmen sollen.
- Nun werden durch Inkludieren von bird.inst.conf die für alle beteiligten IGP-Instanzen (dieser konkreten IGP-Kopplung) relevanten Definitionen getätigt. Vor allem die Listen der Präfixe, die (ggf. samt Subnetzen) in dieser IGP-Kopplung überhaupt geroutet werden (“Routed”) bzw. auf keinen Fall geroutet werden (“Unrouted_global”). Diese werden von der Funktion fn_a_acceptable_routes ausgewertet werden. (Für Experimente mit cold/hot potato Routing müssen außerdem alle (benachbarten) IGP-Instanzen wissen, ob ein bestimmter Nachbar hot potato verwendet, weil dann keine Transitrouten von ihm angenommen werden dürfen.)
- Es folgen teils aufeinander aufbauende Funktionsdefinitionen, die in den Import- und Exportfiltern der verschiedenen Protokollinstanzen verwendet werden. In diesen Funktionen findet die Mechanik der IGP-Kopplung statt. Eine Detailbesprechung würde hier zu weit führen, aber zur Verständnishilfe sei empfohlen, sich anhand der obigen Beschreibung der Schleifenverhinderung die jeweiligen Filterfunktionen gemäß ihres oben beschriebenen Sinnes zu erschließen. Ein eigenes Thema bildet der o.g. workaround, dem (leider) die Verwendung von iBGP samt zugehöriger Filter zu verdanken ist.
- Zu guter Letzt wird bird.inst.local.conf inkludiert, von wo aus die Konfigurationen der lokalen OSPF-Instanzen (bird.instX.conf, bird.instX.local.conf) inkludiert werden, also aller Instanzen, an denen dieser Grenzrouter teilnimmt. Die Filter dieser Protokollinstanzen verwenden die im vorigen Schritt definierten Funktionen zur Filterung und Manipulation der Routen.

Der Vollständigkeit halber sei nochmal erwähnt, dass die (OSPF-)Router innerhalb einer Instanz nichts über die IGP-Kopplung wissen müssen. D.h. dort ist keine spezielle Konfiguration erforderlich, nur die für die Teilnahme an ihrer Instanz nötige. Die entsprechenden Konfigurationen (bei OSPF: area Definitionen) befinden sich auf den Grenzroutern in bird.instX.local.conf, aber eben für mehr als ein X, d.h. für mehrere Instanzen.

code walk

Eine detailliertere Besprechung der Konfiguration sollte an den entsprechenden Stellen der Konfigurationsdateien erfolgen. Dies ist bisher erst ansatzweise der Fall. TBD.