Beyond CIDR Aggregation

Patrick Verkaik*, Andre Broido*, kc claffy*,
Ruomei Gao***, Young Hyun*, Ronald van der Pol**

*CAIDA, San Diego Supercomputer Center,
University of California, San Diego, CA.
{patrick, broido, kc, youngh}@caida.org

**NLnet Labs,
Amsterdam, The Netherlands.
{rvdp}@nlnetlabs.nl

***College of Computing,
Georgia Institute of Technology, Atlanta, GA.
gaorm@cc.gatech.edu

Abstract—
Previously Broido and Claffy analysed the global Internet interdomain routing system based on BGP policy atoms: equivalence classes of prefixes based on common AS path as observed from a number of topological locations [6] [7]. In this report we define a variant of policy atoms, called *declared atoms*. Declared atoms constitute an aggregation mechanism complementary to CIDR aggregation. We describe a new routing architecture, called atomised routing, based on BGP and declared atoms. Atomised routing aims for a reduction in the number of routed objects in the default-free zone of the Internet (around 20k declared atoms covering around 113k prefixes), and an improved convergence behaviour of the interdomain routing system. We also demonstrate the viability of incremental deployment of atomised routing.[1]

I. Introduction

Global routing in today's Internet is negotiated among individually operated sets of networks known as Autonomous Systems (AS). An AS is an entity that connects one or more networks to the Internet and applies its own policies to the exchange of traffic. AS policy is used to control routing of traffic from and to certain networks via specific connections. These policies are articulated in router configuration languages and implemented by the Border Gateway Protocol (BGP) [45].

The number of entries in the tables of a BGP router has bearing on both router memory and processor cycles. The number and size of routing update messages tend to increase with the number of prefixes in the RIB (Routing Information Base). These factors affect not only communication costs, but also the CPU resources needed to process updates [26]. Furthermore, while the RIB can be maintained in inexpensive general purpose memory, a copy of the FIB (Forwarding Information Base) is stored in specialised forwarding hardware whose memory is relatively expensive. In addition, reduction of the number of entries in the routing tables is beneficial to infrastructural integrity. Smaller routing tables leave more room to handle an unexpected large influx of routes, e.g. as a result of misconfiguration or implementation errors [13], and also leave more room to make a space/time trade-off within a routing system [15].

Routers in the default-free zone (DFZ) of the Internet carry a large number of *global routes* in their tables, which cover the whole of the reachable IP address space, and are propagated virtually throughout the entire DFZ. To maintain a default-free table, a router must necessarily carry this growing set of global routes, the alternative being that some portion of the Internet is unreachable to it. Therefore the size and dynamics of the DFZ impose requirements on any router that is part of the DFZ, whether the router belongs to a Tier-1 ISP or a small, multihomed customer.

The number of global routes has increased at varying speeds over the years [26], and continues to grow [27]. CIDR (Classless Inter-Domain Routing) [44] [17] was introduced to combat growing routing table sizes and IP address space depletion. Unfortunately the benefits of CIDR are counteracted by disincentives to aggregate [10], leading to the announcement of more specific prefixes in addition to, or instead of, aggregated prefixes.

We introduce the notion of a *declared atom*, an aggregation mechanism complementary to the CIDR aggregate. We describe an architecture based on declared atoms that aims to significantly reduce the number of global routes and routing update messages in the default-free zone of the Internet, and to improve convergence behaviour of the interdomain routing system.

We organise the remainder of this report as follows. Section II provides relevant background information about interdomain routing. Section III defines the notions of computed atom and declared atom. In Section IV we present an overview of the atomised routing architecture, and elaborate details in Sections V to VIII. Section IX discusses the convergence properties of our architecture. The role of the origin AS within the architecture is detailed in Section X. Sections XI to XIV cover practical issues such as incremental deployment, security, tunneling, and how to contain the number of globally routed objects in the real world. In Section XV we discuss our prototype implementation of the architecture. Section XVI presents the analyses we

[1]We gratefully acknowledge support for this work by NLnet Labs and RIPE NCC.

performed to prepare for simulation. Section XVII explores a variation of the declared atom concept, the *provider-declared atom*, which aims at further reduction of the number of globally routed objects. Section XVIII discusses future work. Finally in Section XIX, we conclude by considering advantages and disadvantage of our architecture, in part based on feedback from the community.

II. Background

In this section we provide an overview of the Border Gateway Protocol version 4 [45] [49], and other aspects of interdomain routing relevant to this report. However, we assume the reader is familiar with BGP4 and related standards [44] [17] [12] [51] [3].

A basic BGP exchange consists of an update message announcing (advertising) or withdrawing reachability of a single network prefix via a certain router. The reachability information in an advertisement includes a next hop router, an AS path (a sequence of ASes), and various attributes expressing policy. BGP assumes that (a) the announcement traversed the ASes in the AS path, (b) the advertised prefix can be reached via the next hop, and (c) any packets sent to networks covered by the prefix, through the next hop, will traverse the AS path (in reverse order).[2]

BGP routers maintain BGP sessions with each other, based on TCP, through which they exchange BGP update messages. At the start of a session (e.g. after a previous session has terminated), the two routers exchange an initial set of advertisements based on the routes they know. Subsequently, the routers only exchange incremental updates. Two BGP routers that have a BGP session are called *BGP peers*, and are said to *peer* with one another.

A BGP router maintains an internal *RIB* (Routing Information Base), which associates a network prefix with BGP advertisements received from its BGP peers, and a *FIB* (Forwarding Information Base) that it consults during packet forwarding. The FIB contains a next hop router for each prefix learned through BGP or other (IGP) routing protocols, such as OSPF and IS-IS. BGP applies inbound filtering and local policy decisions to choose the preferred route for each prefix in the RIB and installs that route in the FIB.[3] In addition, the preferred route may be advertised to other BGP peers, subject to loop detection based on the AS path, and (per peer) policy decisions.

Rather than having each BGP router carry routes that cover the entire reachable IP address space, many ASes rely on a *default route*, which typically points to a provider AS. A BGP router that has a default route only carries a subset of routes, e.g. routes for destinations in the local AS, customer ASes, and ASes with which the AS has an AS policy peering relationship[4] [18], and forwards IP packets to other destinations along the default route. An IP packet may be forwarded along several default routes until it reaches a BGP router that has a non-default route covering the destination address of the packet. A smaller number of large, *Tier-1*, ISPs do not have providers that they can rely on for default routing. These ISPs typically have AS peering relationships among one another, and maintain *default-free* routing tables. We refer to the BGP routers in the Internet that maintain default-free routing tables as the *default-free zone* (*DFZ*) of the Internet. The DFZ is not limited to Tier-1 ISPs. For example, a default-free routing table allows other, multihomed ASes to perform outbound traffic engineering more effectively. Also, the edge of the DFZ does not necessarily coincide with AS boundaries. For example, a large ISP may have customer access routers with default routes pointing to distribution[5] or core routers that carry default-free routing tables. In addition, the IGP of an AS in the DFZ does necessarily carry default-free routing tables. For example, an IGP router in such an AS may have a default route pointing to a default-free BGP router in the AS.

CIDR (Classless Inter-Domain Routing) [44] [17] was introduced to combat growing routing table sizes and IP address space depletion. CIDR allows better aggregation of IP address space into variable length IP prefixes. A prefix *addr / p* summarises a contiguous range of IP addresses, the *p* leftmost bits of which match those of *addr*. CIDR is accompanied by a suggested prefix allocation policy that creates opportunities for aggregation. Unfortunately the benefits of CIDR are counteracted by disincentives to aggregate, leading to the announcement of more specific prefixes in addition to, or instead of, aggregated prefixes [10] [6] [9] [26] [30]. In particular, [10] identifies the following causes:

- Multihoming: the practice of announcing a global route through several providers, at most one of which is able to aggregate the announced address space into its own address block.
- Inbound traffic engineering: announcing more specifics of prefixes with non-identical BGP attributes leads to a further splintering of prefixes. This functionality facilitates load-balancing of incoming traffic. Another example is of an AS that avoids paying for traffic destined toward unreachable IP addresses, by announcing to its providers only the parts of an address block that are reachable. We expect this practice to become more common, given the increase of worm activity in the Internet.
- Fragmented address space which cannot be aggregated.
- Failure to aggregate.

Consider the example in Figure 1. AS A has two provider ASes, AS B and C, both of which are attached to the DFZ. Having several providers, AS A is said to be *multihomed*. One reason for multihoming an AS is to improve its connectivity. AS B and C have been allocated the address blocks 3.0.0.0/8 and 4.0.0.0/8, respectively, which they announce into the DFZ. AS A has been allocated an IP address block 3.1.0.0/16 out of the address space of AS B, and a provider-independent address block 192.2.0.0/16. The address blocks are announced to both providers. AS A wishes to balance the load of incoming traffic to 3.1.0.0/16 over the two links. To do so, AS A advertises half of the address block (3.1.0.0/17) to AS B and the other

[2] But note that this assumption does not always hold [29] [37].

[3] In reality, BGP merely makes the preferred route available for the FIB. Whether the route makes it to the FIB depends on the presence of alternative routes preferred by other routing protocols, statically configured routes, etc.

[4] The words 'peer' and 'peering' are commonly used to refer to the settlement-free relationship between ASes, as well as to the relationship between any two neighbouring routers. Henceforth, we will consistently use the terms 'AS (policy) peer' and 'AS (policy) peering' to denote the former sense.

[5] Distribution routers are responsible for aggregating customer routes before propagating them to the core.

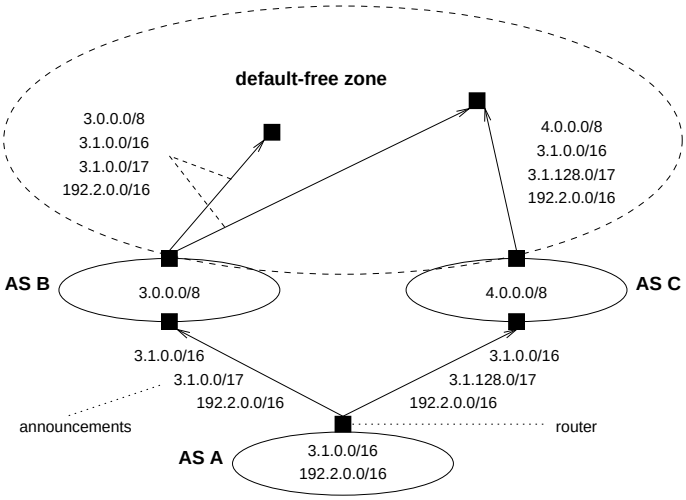

Fig. 1. A multihomed AS.

half (3.1.128.0/17) to AS C. To ensure that the whole of the address block remains reachable should either of its provider links fail, AS A additionally advertises the entire address block (3.1.0.0/16) to both providers. Since the two more specific advertisements take precedence, this approach achieves load balancing.

The prefixes advertised to AS C cannot be CIDR-aggregated into AS C's own prefix advertisement, and must therefore be advertised separately into the DFZ. AS B could aggregate two prefixes it received from A, 3.1.0.0/16 and 3.1.0.0/17, into AS B's own prefix advertisement. But this approach would cause all traffic destined for AS A to be attracted toward the more specific prefixes advertised by AS C, defeating the load balancing objective. Therefore AS A convinces AS B to announce all three prefixes into the DFZ.

Table I shows the tails of the AS paths for the prefixes in Figure 1.

Prefixes	AS Paths (Tail)	Computed Atoms
3.1.0.0/16 192.2.0.0/16	B-A & C-A	A1
3.0.0.0/8	B	A2
4.0.0.0/8	C	A3
3.1.0.0/17	B-A	A4
3.1.128.0/17	C-A	A5

TABLE I
AS PATHS AND COMPUTED ATOMS DERIVED FROM FIGURE 1.

III. ATOMS

Broido and Claffy introduced the notion of *(BGP policy) atom* [6] as a means to analyse the complexity of the interdomain routing system. By analysing a number of Route Views [50] peers through atoms, the authors found that a renumbering of the Internet address space could potentially reduce the size of a complete DFZ BGP table at that time by a factor of two while preserving all globally visible routing policies [7]. In addition they found that the number of atoms properly scales with the Internet routing system's growth.

In this report we distinguish two kinds of atoms: *computed atoms*, which correspond to the atom concept in [6] and are used for analysis, and *declared atoms*, which we introduce as a more practical alternative on which to base a routing architecture.

A. Computed Atoms

A computed atom is defined relative to a number of observed BGP routers as follows. Two prefixes are said to be *path equivalent* if no BGP router can be found among the observed BGP routers that sees the two prefixes with different AS paths.[6] An equivalence class of this relation is called a *computed atom*. This definition implies that prefixes in the same atom share a set of AS paths.

Table I shows computed atoms for the prefixes in Figure 1, as observed by the two routers shown within the core of the DFZ. The prefixes 3.1.0.0/16 and 192.2.0.0/16 are observed with the same AS paths by both routers and are therefore part of the same atom. All other prefixes have unique AS paths.

We estimate the number of computed atoms for the Internet using interdomain BGP routing tables obtained from the University of Oregon's Route Views project [50]. Route Views runs a number of *route collectors*, that peer with BGP routers, called *Route Views peers*, located in several ASes. Route Views makes periodic BGP IPv4 RIB dumps of one of its route collectors (*route-views.oregon-ix.net*) publicly available. Another collector, *route-views2.oregon-ix.net*, provides not only BGP IPv4 RIB dumps but also the BGP update messages received from its peers. Using the table dumps and updates from these collectors we constructed an *8 hour* and a *5 day* dataset.

Dataset	Start time	End time	Peers
8 hour	Jan. 15, 2003 04:01 PST	Jan. 15, 2003 12:03 PST	35
5 day	Jan. 15, 2003 00:00 PST	Jan. 20, 2003 00:10 PST	14

TABLE II
DATASETS USED.

For the *8 hour* dataset (Table II) we used two table dumps from *route-views.oregon-ix.net*. Of the 61 Route Views peers that contribute to the table dump, we selected at most one peer per AS, and only those peers that carried a full routing table (consisting of at least 110,000 prefixes for this dataset), resulting in 35 peers. We use full routing tables in this analysis to avoid measurement anomalies [57] and measurement bias. We only used prefixes observed by all 35 peers.

We created the *5 day* dataset by taking an initial table dump from *route-views.oregon-ix.net* and by running updates from *route-views2.oregon-ix.net* against it to construct the final snapshot. The update stream starts at 00:00:40 on Jan. 15, 2003, and ends at 00:10:00 on Jan. 20, 2003.[7] For this dataset we narrowed down the peer selection from 35 peers (determined as for the *8 hour* dataset) to 14 peers, by selecting only those peers

[6] After removing consecutive duplicate ASes (prepending) from AS paths.
[7] Ideally, the table dumps and the updates should both be taken from the same route collector.

whose updates we observed in the update stream. In contrast with the *8 hour* dataset, we used all prefixes observed at one or more of the 14 peers.

Dataset	Prefixes	Computed Atoms	Recurrence
8 hour	113k	30k	95.6%
5 day	123k	27k	89.7%

TABLE III
COMPUTED ATOMS.

For the *8 hour* dataset we computed a total of 30k atoms covering 113k prefixes, both in the initial and the final snapshot. We define the *recurrence ratio* as the percentage of atoms present in the initial snapshot that are also present in the final snapshot. The recurrence ratio for the *8 hour* dataset is 95.6%. Note that this statistic does not imply that 95.6% of atoms were stable during that period. Rather, it is an indication of the long-term persistence of a grouping of prefixes in the routing system. Table III summarises the statistics for the computed atoms in both datasets.

We note that Route Views provides only a limited view of the interdomain routing system. Mostly customer-provider relationships are observable, while AS peering relationships are often not captured [8]. Increasing the number of observed Route Views peers improves coverage. However [7] showed that for May 2001 data, 90% of the atoms computed from 27 peers were produced by limiting the selection to the 8 largest peers.

In BGP, inbound traffic engineering and export policies are expressed by (i) the act of announcing a route, (ii) prepending (inserting extra copies of an AS in the AS path), (iii) communities [12], and (iv) multi-exit discriminators (MEDs). Communities and MEDs cannot be observed more than one hop away from the AS that applied them, yet they affect propagation and acceptance of an announcement, and ultimately the set of AS paths via which Internet routers will observe it. In other words, although AS paths (and therefore computed atoms) only implicitly reflect policies of ASes, it is likely that most policy information is present in the fact of acceptance and propagation of an announcement. This observation is supported by the fact that on March 1, 2003 the number of atoms computed based on prepended paths is only 1% larger than the number of atoms based on paths from which prepending is removed.

B. Declared Atoms

Computed atoms are useful for analysing the complexity of the interdomain routing system, but do not lend themselves well to routing. A routing protocol must respond quickly to e.g. link failures, and recomputing atoms for such events is likely to take too long. In particular, the computation involves the observation of multiple, potentially distant, routers.

We now introduce a second type of atom, more amenable to routing, which we call a *declared atom*. As the name suggests, this type of atom is declared by an AS, rather than empirically observed. In an atomised routing architecture, an AS groups prefixes that it deems equivalent into a declared atom. It then announces this declared atom, instead of the prefixes, to other

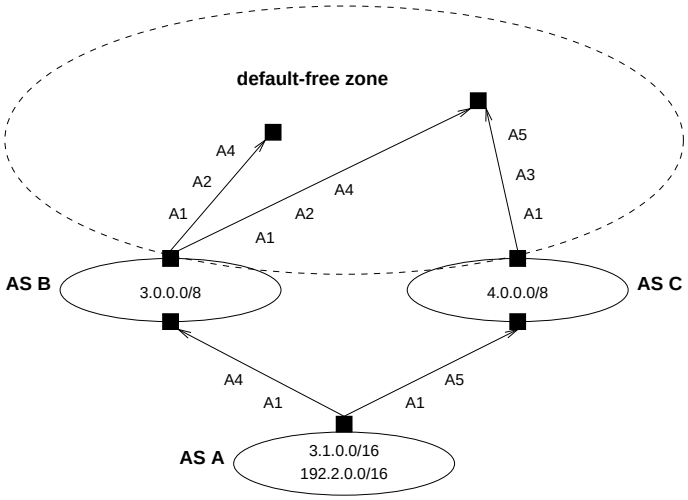

Fig. 2. Origin-declared atoms for Figure 1.

ASes. Essentially a declared atom is similar to a CIDR aggregate, without the restriction that the aggregate must form a contiguous address block.

The natural place to declare an atom is at the AS that originates the prefixes of the atom (the *origin AS*). This report mostly considers such *origin-declared atoms* and unless otherwise noted we use the term *declared atom* to denote an *origin-declared atom*. For the example shown in Figure 1, the (minimal) set of declared atoms coincides with the set of computed atoms in Table I. Figure 2 shows the corresponding announcements made by each of the ASes.

Under declared atoms, other ASes must accept prefix groupings made by declaring ASes. For example, AS B in Figure 2 cannot apply different policy to the prefixes in atom A1 declared by AS A. At first this restriction seems limiting, but empirically, 85% of differentiation (in terms of AS paths) among prefixes today is observed between the origin AS and its adjacent ASes [1].

C. Estimating the Number of Declared Atoms

Dataset	Prefixes	Comp. Atoms	Decl. Atoms	Recurrence
8 hour	113k	30k	20k	97.8%
5 day	123k	27k	21k	93.4%

TABLE IV
ESTIMATED ORIGIN-DECLARED ATOMS.

We can estimate the number of declared atoms corresponding to prefixes found in today's global routing tables by counting the number of distinct sets of origin links observed for prefixes. An *origin link* of a prefix is the origin AS and the first hop AS of one of the prefix's AS paths. For example the set of origin links observed for prefix 3.1.0.0/16 (Figure 1) is {B-A, C-A}. Prefix 192.2.0.0/16 shares the same origin link set. To estimate the number of declared atoms we assume that prefixes with identical origin link sets are placed in one declared atom by the origin AS. For the *8 hour* dataset, we arrive at total of 20k distinct

origin link sets (both in the initial and final snapshots). Thus the estimate for the number of declared atoms is 20k, versus the 30k computed atoms we derived earlier (Table IV). We associate with each origin link set a *prefix set*, i.e. the set of prefixes that share that origin link set. When we compare the prefix sets in the initial and final snapshots, 97.8% of the prefix sets in the inital snapshot are present in the final snapshot. The number of declared atoms can theoretically be further reduced if we allow them to be declared further away from the origin as discussed in Secion XVII. Table IV summarises the statistics for origin-declared atoms in both datasets.

Note that this estimate relies on two assumptions. First we assume that an *observed origin link set* corresponds to an *actual origin link set* one might observe at the origin AS. Second, we assume that the *actual prefix set*, i.e. the prefix set associated with the *actual origin link set*, is a set of prefixes that the origin AS would declare as a unit.

• *Assumption 1* There are several ways in which the view of the Internet is distorted by Route Views. First, Route Views does not offer a complete picture of the Internet, and it is possible that some actual origin links are never observed by Route Views. This phenomenon tends to decrease the number of observed origin link sets. Another source of distortion consists of events that occur between the origin AS and the point of observation. For example, if a particular origin link is observed as part of the AS path of a single route and that route is withdrawn due to disrupted connectivity upstream of the origin link, then the origin link will disappear from view. Another reason that an origin link might disappear from view is that one of the routers upstream of the origin link preferred a different route whose AS path contained a different origin link. A third significant source of distortion is the convergence behaviour of BGP. For some routing events, BGP can take over an hour [36] to converge.
Even if we ignore the problem of hidden origin links, we cannot simply assume that every observed origin link set corresponds to an actual origin link set, since BGP convergence behaviour may cause spurious origin link sets to be observed. However, again ignoring the hidden origin link problem, one may expect that a subset of observed origin link sets corresponds to actual origin link sets. In particular, it is likely that an actual origin link set that is stable for a long period of time will appear as an observed origin link set. We base our estimate of the number of declared atoms on this assumption. The high recurrence ratio (Table IV) between two snapshots separated by a period well over an hour increases our confidence that the effects of convergence are negligible for this particular measurement.
• *Assumption 2* Our second assumption is that the actual prefix set is a set of prefixes that the origin AS considers to be a unit. This assumption may be wrong, since an origin AS may want to group its prefixes with finer granularity than origin link sets. For example, AS A in Figure 1 may wish to place prefixes 3.1.0.0/16 and 192.2.0.0/16 in separate declared atoms rather than group them together as in Table I. We return to this issue in Section XIV.

The above assumptions imply that we should treat the estimate of the number of declared atoms as a lower bound.

IV. Atomised Routing Architecture

BGP operates on the level of individual prefixes. Each table entry and route computation is based on a single prefix as the basic element. The ability to pack together multiple prefixes in a BGP update message [45] is a considerable improvement but does not reduce the number of routed objects in the DFZ, nor does it eliminate per-prefix processing of BGP updates. Furthermore, this technique can only be applied to prefixes with identical attributes. For example, announcements of prefixes with different origin ASes cannot be part of the same BGP update. In this section we propose a new routing architecture based on BGP and declared atoms, which we call *atomised routing*, the main features of which are:

- a reduced number of routed objects in the DFZ
- potential for improved convergence behaviour
- incrementally deployable
- applicable to IPv4 as well as IPv6[8]

We first give an overview of the atoms architecture, and elaborate details in subsequent sections.

In our architecture, an atom is declared (Section III-B) as a container of a set of prefixes that appear throughout the DFZ today, i.e. are not CIDR-aggregated away. We call a prefix that is part of a declared atom an *atomised prefix*. We identify an atom by an IPv4 prefix,[9] which we call an *atom ID*, and which is drawn from the regular IPv4 prefix space. As we will see, this allows atoms to be routed by unmodified BGP routers. Atomised prefixes can be more specifics of other atomised prefixes (and maintain today's semantics of specificity), possibly in different atoms. An atom ID, however, is neither a more nor a less specific of any other atom ID or atomised prefix. The atoms architecture focuses on reducing the number of BGP-routed objects in the DFZ and distinguishes the inside of the DFZ from the rest of the Internet. Within the DFZ, an atomised prefix inherits the routing attributes from the atom that it is part of. To ensure that the routing attributes of an atomised prefix are well-defined, an atomised prefix should not be declared part of more than one atom. In particular, there is no hierarchical relationship among atoms, in terms of the prefixes they contain. However, during convergence and as a result of misconfiguration, it is inevitable that atoms occasionally *overlap* (share one or more atomised prefixes). For these cases, our architecture defines a procedure that resolves overlapping atoms.

Figure 3 highlights the roles that different interdomain routers play in our architecture, namely:

• *Edge routers (E)*, which are DFZ routers that forward IP packets among DFZ and non-DFZ routers, and appear at the edge of the DFZ.
• *Transit routers (T)*, which are DFZ routers that merely forward packets among other DFZ routers. These are atoms-unaware BGP routers.
• *Atom originators (O)* are routers outside the DFZ that declare atoms and announce BGP routes for atoms IDs.[10]

[8] However, the scope of this report is IPv4.
[9] An IPv6 prefix can also serve to identify an atom. In this report we assume IPv4 prefixes are used.
[10] Note that in principle an edge router could also declare and announce atoms. For clarity, in this report we exclusively assign this task to the atom originator role.

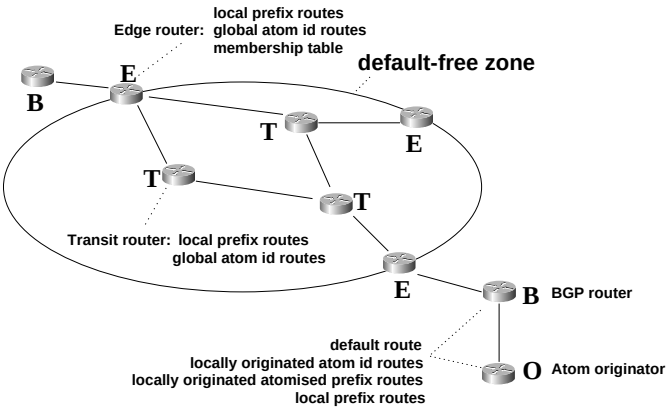

Fig. 3. Roles and routes in the atoms architecture.

• Other *BGP routers (B)*, which are atoms-unaware BGP routers outside the DFZ.

For the moment we assume that the role of edge routers is performed by access or distribution routers in an ISP network, and the role of transit routers is performed by core routers.

As shown in Figure 3, each AS inside or outside the DFZ contains BGP routes for *local prefixes* that are routed within the AS (and possibly a limited number of nearby ASes), but are not globally routed. We will mostly ignore local prefix routes. BGP routes for *global atom IDs* appear throughout the DFZ. Outside the DFZ, a global atom ID route typically only appears in its origin AS, and possibly a limited number of ASes near the origin AS (*locally originated atom ID routes* in Figure 3). Note that Figure 3 does not show *global prefix routes*. In our architecture we place today's global prefixes inside atoms as *atomised prefixes*. Atomised prefixes do not have BGP routes inside the DFZ. However outside the DFZ, BGP routes for atomised prefixes may appear, but only in areas where the corresponding atom ID route also appears. Therefore, an atomised prefix route typically only appears in (or near) its orgin AS (*locally originated atomised prefix routes* in Figure 3).

Our architecture is composed of three main functions: *atom-based forwarding* (Section V), *atom routing* (Section VI), and *atom membership* (Section VII). Edge routers (Section VIII) play a special role in all these functions.

Atom-based forwarding is an encapsulation mechanism that allows IP packets to be forwarded based on atom IDs. An IP packet that needs to traverse the DFZ is encapsulated by an edge router to form a packet with the atom ID as the destination IP address.[11] The packet is forwarded through and out of the DFZ to the destination AS based entirely on the atom ID destination address and atom ID routes inside and outside the DFZ. As the packet reaches the atom originator at the destination AS, it is decapsulated and subsequently forwarded based on the original destination IP address and prefix routes.

Atom routing is BGP applied to atom IDs and atomised prefixes. In our architecture, routers inside and outside the DFZ route atom IDs in the same way that routers today route global prefixes. In addition, atom routing is responsible for ensuring that atomised prefix routes are present in selected (see above) areas outside the DFZ, and preventing atomised prefix routes from entering the DFZ (Figure 3). Therefore edge routers filter atomised prefix routes to prevent them from entering the DFZ, and selectively announce atomised prefix routes toward routers outside the DFZ.

The *atom membership protocol* distributes a mapping between atom IDs and atomised prefixes as declared by atom originators. Edge routers receive atom membership information from atom originators, and distribute the information among one another, bypassing the transit routers of the DFZ. As a result, the transit routers of the DFZ never see atomised prefix routes. The atom membership information is stored in membership tables (Figure 3).

V. Atom-Based Forwarding

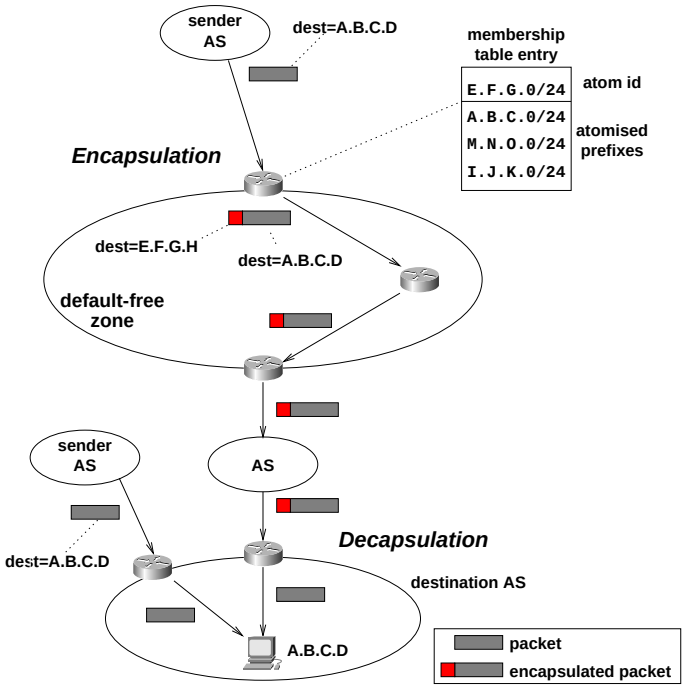

Fig. 4. Atom-based forwarding.

As discussed above, routers outside the DFZ carry BGP routes for local and atomised prefixes as well as for atom IDs. However inside the DFZ, routers only carry BGP routes for local prefixes and atom IDs. Therefore inside the DFZ, routers do not have sufficient information to forward packets that have a destination IP address based on an atomised prefix. Our architecture uses encapsulation to enable such a packet to traverse the DFZ, as we now describe. Figure 4 illustrates forwarding on prefixes and atom IDs. A packet originating outside the DFZ is initially forwarded based on prefix routes. If it reaches its destination without entering the DFZ, it never gets encapsulated. However if the packet does enter the DFZ, the ingress edge router of the DFZ encapsulates it before further forwarding, even if the edge router is the only DFZ router the packet traverses. From then on the packet is forwarded based on atom ID routes until it reaches the atom originator in the destination AS, where the atom orig-

[11] Technically, the destination is an IP address *based on* the atom ID, since the atom ID is a prefix.

inator decapsulates it. Note that, in order to avoid forwarding loops (Section IX), the packet is *not* decapsulated when it leaves the DFZ. The ingress edge router effectively tunnels the packet to the atom originator. If a packet needs to traverse the DFZ more than once (e.g. due to a routing anomaly), only the first ingress edge router encapsulates. Edge routers must therefore be able to tell whether a packet has been encapsulated. A packet originating at an edge router is immediately encapsulated. Packets originating at a transit router must be forwarded to a nearby edge router for encapsulation. Apart from encapsulation and decapsulation there are no changes to forwarding behaviour.

VI. Atom Routing

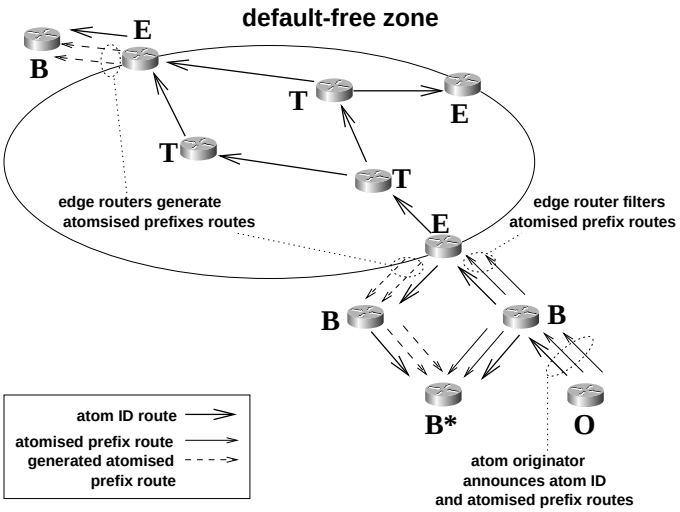

Fig. 5. Atom routing.

Atom routing is responsible for routing atom IDs and atomised prefixes in BGP as indicated in Figure 3. In our architecture, routers inside and outside the DFZ route atom IDs in the same way that routers today route global prefixes, and apply similar policies. Therefore just as a global prefix today appears throughout the DFZ and additionally may appear outside the DFZ in selected areas (typically near the origin AS of the global prefix), so an atom ID appears throughout the DFZ as well as in selected areas outside the DFZ. In contrast, atomised prefixes never appear inside the DFZ, and outside the DFZ are present only in areas where the corresponding atom ID appears.

Figure 5 illustrates atom routing for an atom containing two atomised prefixes. (The figure does not show local prefix routes.) The process begins when an atom originator (O) announces an atom ID. Outside the DFZ both the atom ID and its atomised prefixes are routed (as shown in Figure 3). Therefore the originator announces BGP routes for the atom ID as well the atomised prefixes. Inside the DFZ, only the atom ID is routed (Figure 3); the edge router therefore *filters* the atomised prefix routes that it receives,[12] but propagates the atom ID route into the DFZ. When an edge router receives an atom ID announcement (from within or outside the DFZ), it *generates* BGP routes for the atomised prefixes to routers outside the DFZ, but only if its policy allows it to propagate the atom ID route there. Generally one would not expect global routes such as an atom ID to propagate outside the DFZ, except in the area where the global route was originated. Section VII discusses how the edge router knows what atomised prefix routes to generate.

To allow edge routers to easily filter atomised prefix routes, we define a new optional transitive BGP attribute [45] which acts as a marker for atomised prefix routes. The *atomised marker attribute* does not contain any information: its mere presence suffices. Every atomised prefix route shown in Figure 5 carries the marker attribute. An atom originator attaches the marker to atomised prefix routes it orginates. Similarly, an edge router attaches the marker to atomised prefix routes it generates.

Atom ID routes and atomised prefix routes are subject to ISP policy, just as prefix routes are today. In Figure 5, the atom originator and other BGP routers outside the DFZ apply the same policy to the atom ID and its atomised prefixes. However, our architecture does not require policy for atom IDs and atomised prefixes to be configured consistently, either in the atom originator or in other BGP routers. In particular this flexibility opens the opportunity for the originator to engineer inbound traffic that is originated 'nearby' differently from traffic originated further away. We return to the capability of differentiating local and global policy in Section XIV.

When an edge router generates an atomised prefix route from an atom ID route, it bases the BGP attributes of the atomised prefix route on those of the atom ID route, including the AS path.[13] In addition, it attaches the atomised marker attribute as discussed above. Therefore some areas outside the DFZ may have atomised prefix routes originated by the atom originator, as well as atomised prefix routes generated by edge routers for the same prefixes. For example in Figure 5, router B* receives atomised prefix routes originated by O (solid thin arrows) as well as generated atomised prefix routes (dashed arrows). The generated routes may have different attributes from the originated routes since the former are based on the attributes of an atom ID route. This situation is no different from today's, where each BGP router may modify, add, and drop BGP attributes before propagating a route. However, the presence of generated atomised prefix routes carrying the atom ID route's attributes may subvert an atom originator's deliberate policy to attach different attributes to atom ID routes and atomised prefix routes.

VII. Atom Membership

The atom membership protocol is an overlay protocol responsible for conveying atom declarations from atom originators to an edge router, and for disseminating this information from there to all other edge routers. Each AS in the DFZ contains one or more edge routers. An atom originator *declares* atoms by partitioning its prefixes into sets, assigning an atom ID to each set, and sending the atom IDs and sets of atomised prefixes to an edge router. After an atom originator has declared an atom in this way, it can issue updates to the atom by redeclaring it with a modified set of atomised prefixes.

[12] Filtering atomised prefixes somewhat resembles the common practice of filtering routes for prefixes that are longer than /24.

[13] An edge router filters atomised prefix routes on ingress.

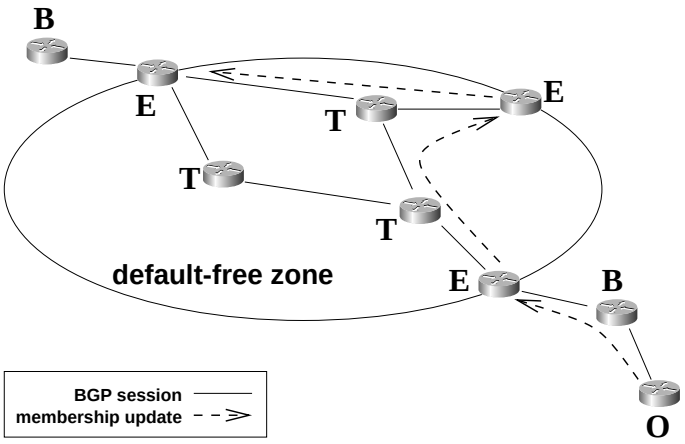

Fig. 6. Atom membership.

Edge routers store this information in an *atom membership table*. This table lists, for each atom ID, the list of atomised prefixes in the atom (i.e. an $atom\ ID \leftrightarrow prefix\ set$ mapping), as well as several other attributes. Figure 6 gives a high level view of the atom membership protocol. The important thing to notice is that the protocol sends the membership messages only to edge routers (E). It bypasses BGP (B) and transit (T) routers. Although these routers forward membership messages (as they do any other IP packet), they do not process the messages. Thus the atom membership protocol and its dynamics do not incur CPU or memory load on BGP and transit routers, nor is the propagation of membership messages delayed by processing in these routers.

The atom membership protocol is not a typical routing protocol in that it does not perform route computation. Rather, it distributes among edge routers the atom membership table, which, like DNS, is independent of any location in the Internet: any edge router will converge to an identical $atom\ ID \leftrightarrow prefix\ set$ mapping. In addition, the contents of a membership update are independent of which router's neighbour sent the update. BGP does not have this independence property, so a BGP router must remember for each neighbour all the routes currently advertised by that neighbour in a *RIB-In* [45] table. In contrast, an edge router may discard received membership updates once they have been processed.

Membership messages are carried by TCP sessions between edge routers and atom originators. Each edge router and atom originator is configured with a number of neighbours and maintains a (multihop) session[14] with each of them, which we call a *membership session*. For example, the membership updates in Figure 6 (dashed arrows) are each carried by a membership session. As is the case for a BGP session, a table exchange takes place at the start of a membership session between two edge routers, and subsequent update messages carry incremental updates. Similarly, in the case of a membership session between an atom originator and an edge router, the atom originator sends all its declared atoms to the edge router at the start of the session and subsequently sends incremental updates. However, the edge router never propagates updates to the atom originator.

In EBGP (exterior BGP), multihop sessions are normally avoided[15], due to the increased likelihood of session resets relative to single-hop sessions, combined with the potentially widespread damage a session reset may incur [53]. The reason that a multihop session can have such a widespread effect in BGP is that BGP maintains reachability of destinations through paths, and both peers are required to interpret a session reset as unreachability of destination prefixes through paths traversing the other peer, and propagate this unreachability information to other peers. The atom membership protocol, on the other hand, does not maintain reachability of prefixes through paths. Initially, a session reset has no effect on the two edge routers, other than to delay propagation of membership updates. When the session is reestablished, a table exchange takes place, but the effects of this exchange do not spread beyond the two edge routers directly involved.[16]

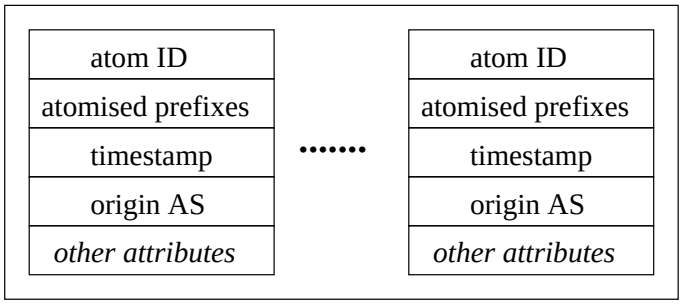

Fig. 7. BGP atom membership message.

Figure 7 depicts the contents of an atom membership message. Each message may carry updates for multiple atoms. An update for an atom contains the atom ID, the atomised prefixes declared to be part of the atom, a timestamp, the AS of the atom originator, and optionally other attributes. In Section IX, we discuss the semantics of updates for multiple atoms per message. In this section we assume a message carries a single update.

Membership updates for an atom may reach an edge router through multiple paths, and can arrive out of order or be received more than once. To allow the original order to be restored and duplicates to be eliminated, each update carries a timestamp that acts as a version number for the atomised prefix set of an atom. The timestamp provides a unique ordering of updates to an atom and is defined by the atom originator.[17] However, since each update carries the full set of prefixes of an atom, an edge router is not required to process every update, nor to maintain a reordering buffer. The edge router may opportunistically process updates as they arrive, so long as the timestamps of the processed updates increase monotonically for a given atom. In principle, an edge router discards updates carrying a timestamp older than, or equal to, the timestamp of the last update processed for that atom. However, the details are a little more intricate, as we de-

[14] A multihop session is a TCP session between two routers that spans multiple sequential links.

[15] IBGP (interior BGP) multihop sessions are common.

[16] Other than propagating membership updates that were delayed while the session was down.

[17] In this section we assume that the declaring AS has a single atom originator. In the case of multiple atom originators per AS, creating such a unique ordering is non-trivial. Section X discusses this further.

scribe next.

The membership protocol described above could be implemented by flooding updates over all membership sessions, ignoring updates that do not carry a new timestamp for the atom. However, such unconstrained propagation may lead to customer ASes propagating updates among their providers, and AS peers propagating to each other updates from their providers and other AS peers. Although this does not harm the integrity of the membership protocol, ISPs and their customers have no interest in propagating routing updates along these paths, and may find unconstrained propagation undesirable. Therefore, we constrain the paths that updates are allowed to propagate along, while still guaranteeing that all edge routers receive the updates they require, as follows.

An edge router or atom originator labels each membership session it maintains with a router in another AS[18] with an attribute describing (a) the policy relationship its AS has with its peer's AS [18], and (b) whether the peer is an edge router or atom originator. The *peer label* has one of the following values:

- *AS Peer* — a router that labels a session as *AS Peer* and the router at the other end of the session are both edge routers, and their ASes are AS policy peers of one another. We call this membership session an *AS peering session*.[19]
- *Provider* — a router CR that labels a session as *Provider* and the router PR at the other end of the session are both edge routers. Router PR is in an AS that is a provider of CR's AS and must label its session as *Customer*. We call this membership session a *customer-provider session*.
- *Customer* — a router PR that labels a session as *Customer* and the router CR at the other end of the session are both edge routers. Router CR is in an AS that is a customer of PR's AS and must label its session as *Provider*. The membership session is a *customer-provider session*.
- *Edge* — a router CR that labels a session as *Edge* is an atom originator and the router PR at the other end of the session is an edge router. Router PR is in an AS that is a provider of CR's AS and must label its session as *Originator*.
- *Originator* — a router PR that labels a session *Originator* is an edge router and the router CR at the other end of the session is an atom originator. Router CR is in an AS that is a customer of PR's AS and must label its session as *Edge*.

At the start of a session, membership peers exchange their labels for the session. Using this techique of *peer labeling* together with an exchange of peer labels at session establishment, an edge router or origin AS router is able to detect inconsistencies between its own and its peer's configuration of a membership session. Peer labeling and verification of peer labels increases the level of robustness, since for misconfiguration to occur at least two adjacent ASes must misconfigure.[20] The most straightforward way to label the membership sessions between routers is to follow the actual business relationships between the ASes that the routers belong to. However, technically it is possible to diverge from business relationships. Indeed, many business relationships do not fall strictly into either category of Provider/Customer or AS Peering [40]. Also, we have not covered backup relationships.

Propagation of membership updates by an edge router then proceeds in accordance with a number of rules that resemble those in [18]:

1. New membership updates, i.e. updates carrying a timestamp that the router has not seen before for the atom, from an *Originator* or *Customer* are propagated to all (other) edge routers.
2. New membership updates from a *Provider* or *AS Peer* are propagated to all *Customer* edge routers.
3. If a membership update *U2* is received from an *Originator* or *Customer C* that carries the same timestamp as the last membership update *U1* received for that atom, and if *U1* was received from an *AS Peer* edge router *AP*, then *U2* is propagated to all *Provider* and *AS Peer* edge routers, excluding *AP*. We explain this rule in more detail below.
4. If a membership update *U2* is received from an *Originator* or *Customer C* that carries the same timestamp as the last membership update *U1* received for that atom, and if *U1* was received from a *Provider* edge router *P*, then *U2* is propagated to all *Provider* and *AS Peer* edge routers, including *P*. We explain this rule in more detail below.

Note that with these rules an edge router never propagates updates to an atom originator.

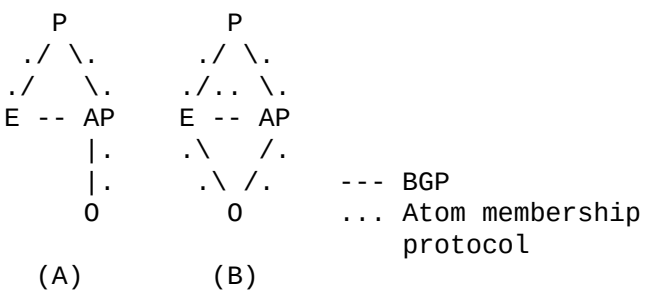

Fig. 8. Examples of structured membership propagation.

Under normal circumstances, membership AS peering sessions are not necessary: customer-provider sessions are sufficient for global distribution of updates. For example, in Figure 8A, membership updates from *O* propagate through *AP* and *P* to *E*. However, if connectivity between *AP* and *P* is disrupted, BGP updates from *O* continue to propagate from *AP* to *E* through the AS peering link; yet *E* does not learn of membership updates from *O*, since there is no membership AS peering session between *AP* and *E*. Therefore, ASes that have a BGP relationship should have a corresponding membership session.

Figure 8B illustrates the need for Rule 3. Consider a membership update from *O* that propagates through *AP* to *E* before it is able to propagate directly from *O* to *E*, e.g. because the membership session between *O* to *E* was temporarily disrupted. Without Rule 3, if the session between *AP* and *P* is disrupted, *P* does not learn of the update, since *E* will not propagate a membership

[18] Intra-AS sessions are discussed below.

[19] See footnote 4.

[20] We are evaluating applying the peer labeling technique to BGP. Note that such a technique does not comprehensively attack the general problem of conflicting policies in BGP [19]. In particular, the technique does not encompass verifying consistency of the peer label with the internal policy of the AS. Nor does it detect inconsistencies that can only be detected by examining the policies of more than two ASes. However, our technique's advantage lies in its simplicity and the fact that it can be applied without a central registry.

update from an *AS Peer* to a *Provider*. Rule 3 ensures that when *E* receives the update directly from *O*, *E* still propagates it to *P*, even though the update does not carry a new timestamp. A similar case applies for Rule 4. Note that Rule 4 also propagates a customer-received update to the provider from which the last update was received, allowing that provider to apply Rule 3 or 4 in turn.

We briefly summarise the intra-AS membership protocol, i.e. the case of an AS containing multiple edge routers. For the intra-AS membership protocol, we are not interested in avoiding certain propagation paths between edge routers (as we are for the inter-AS case), so we allow the edge routers to flood membership updates through their AS. As in the inter-AS case, an edge router uses the timestamp of a membership update to detect duplicates and reordering of updates. The intra-AS membership topology can be arranged as a full mesh or, for better scalability, in a route-reflector-like hierarchy [3]. An edge router E in AS A that receives an update from a router R in another AS attaches an additional attribute to the membership update before propagating it through AS A. The attribute contains E's label for the membership session between E and R, precisely as defined earlier (one of *AS Peer*, *Provider*, *Customer*, or *Originator*), and is not sent outside AS A. This way other edge routers in AS A can apply the above propagation rules when sending to other ASes. Note that Rules 3 and 4 require an update U to be propagated through AS A twice in the following case: the first instance of U was received from a router in a provider or peer AS of A, and the second instance of U (carrying the same timestamp) was received from a customer AS of A.

Apart from propagating membership updates, an edge router performs additional processing to update its data structures, resolve conflicts between overlapping atoms, and generate atomised prefix BGP routes toward routers outside the DFZ. We discuss this additional functionality in Section VIII.

VIII. Edge Router

The edge router plays a central role in all functions of the atoms architecture: atom-based forwarding, atom routing, and atom membership. This section presents details of the internal organisation of an edge router.

A. Encapsulation

The edge router's task in atom-based forwarding is encapsulation of IP packets (Figure 4). In addition to a forwarding table, an edge router maintains an *encapsulation table* that maps an IP address to an atom ID. Specifically, if an IP address ip is part of atomised prefixes $p_1, \ldots, p_n$, and p_i is the most specific prefix among $p_1, \ldots, p_n$, then the encapsulation table maps address ip to an atom a, such that $p_i \in a$. The algorithm for encapsulation (Figure 9) replaces the existing forwarding procedure. In lines 3-4, the edge router looks up the destination address of the IP packet in the encapsulation table. If no entry exists,[21] the router forwards the packet using the existing forwarding procedure (line 12). If, on the other hand, the encapsulation table contains an entry for the address (lines 7-10), the router encapsulates the packet using Minimal IP-in-IP [42]. The destination address in the new IP header is an arbitrary address picked from the atom ID.[22] The contents of the the remaining fields of the IP header are specified by [42] (not shown). In particular the edge router places its IP address in the source address field. Finally, the existing forwarding procedure forwards the encapsulated packet (line 12). As an optimisation to avoid performing look-ups in two tables, the forwarding table and encapsulation table may be integrated into a single table.

```
01.ip_forward(packet):
02.begin
03.   dest = packet.destination;
04.   atom_id = encaps_table.lookup(dest);
05.   if (atom_id)
06.   begin
07.     insert_header(packet);
08.     atom_dest = pick_address(atom_id);
09.     packet.destination = atom_dest;
10.     packet.source = my_ip_address;
11.   end
12.   old_ip_forward(packet);
13.end
```

Fig. 9. Edge router encapsulation algorithm.

We could use alternative encapsulation protocols to implement forwarding, such as IP-in-IP [41] and GRE [16]. MPLS, if it [47] ever became deployed for interdomain routing, would be another option, requiring only a few modifications to the atomised routing architecture. Indeed, the concepts in our architecture correspond quite well to those behind MPLS (Table V).

Depending on the specific encapsulation protocol used to implement atom-based forwarding, an edge router may be able to determine whether an IP packet has been encapsulated without consulting its encapsulation table. For example, in the case of Minimal IP-in-IP, instead of placing Minimal IP-in-IP's assigned protocol number [42] in the encapsulation IP header, we could request IANA to assign a separate protocol number exclusively for atom-based forwarding, and place the new protocol number in the IP header of an encapsulated packet. An edge router that sees the protocol number in the header of an IP packet knows that the packet has been encapsulated, and need not consult its encapsulation table.

Atomised Forwarding	MPLS
atom	forwarding equivalence class
atom ID	label
encapsulation	initial labeling
forwarding	label swapping

TABLE V
Comparing atoms and MPLS.

We review several issues related to tunnel management [41] in Section XIII.

[21] This covers the case that an IP packet enters the DFZ twice, i.e. $dest$ is (based on) an atom ID.

[22] Recall that an atom ID is represented by a prefix.

B. Membership and Routing

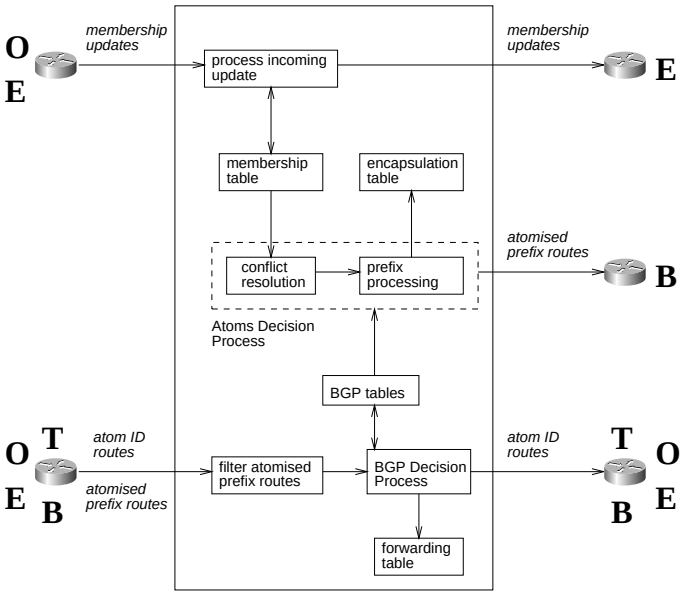

Fig. 10. Flow of data in edge routers.

Figure 10 provides a global picture of how an edge router processes BGP and membership messages, using the same notation as Figure 3. The top of the figure shows the membership updates from atom originators (O) and edge routers (E) arriving at this edge router, and propagation of these updates to other edge routers as described in Section VII. Furthermore, the edge router copies the contents of updates into the membership table. The membership table maps an atom ID to an entry containing (a) a copy of the membership update with the latest timestamp seen so far for this atom (Figure 7), and (b) the identity of the membership peer that sent the update. The identity of the peer is needed for structured propagation (Section VII).

There are no changes to the BGP Decision Process (bottom part of the figure), but note that BGP updates for atomised prefixes are filtered to prevent them from entering the BGP Decision Process (Section VI). We pass atom ID routes to the BGP Decision Process and allow the BGP Decision Process to process them as it processes BGP routes today.[23] The Atoms Decision Process is responsible for (a) resolving conflicts among overlapping atoms, (b) maintaining the encapsulation table, and (c) generating atomised prefix routes in BGP. We describe these functions below.

B.1 Conflict Resolution

We defined declared atoms as a disjoint (non-overlapping) partitioning of prefixes in Section IV. However, because of convergence of the membership protocol following changes to atom declarations, and because misconfiguration is inevitable [35], edge routers can expect to encounter atomised prefixes that have been declared as part of more than one atom.[24] Each edge router independently applies an algorithm to resolve conflicts among overlapping atoms. For all atoms that declare a common atomised prefix, the algorithm picks one of the atoms and assigns the prefix to it. It applies the results of conflict resolution locally, without propagating them to other edge routers (Figure 10). This may lead to inconsistency among edge routes, however, as we explain in Section IX, atom-based forwarding does not depend on edge routers having consistent membership tables and conflict resolution procedures.

In this report we use the resolution algorithm shown in Figure 11. The algorithm prefers atoms with reachable[25] atom IDs to atoms with unreachable atom IDs, and after that, as a tie-breaker, prefers atoms with lower atom IDs. The reason it prefers reachable atom IDs to unreachable atom IDs is as follows. Assume that atoms with lower atom IDs are preferred, regardless of reachability of their atom ID. Now consider an atom originator that becomes permanently unreachable, perhaps as a result of the owner going out of business, and that another atom, declared by another atom originator, subsumes the prefixes of one of the expired originator's atoms. Since the original atom originator has disconnected, it has no way of declaring that the atomised prefixes have been removed from its atom. If its atom ID happens to be lower than that of the successor atom, edge routers that prefer lower atom IDs will permanently (or until garbage collection, see below) associate the prefixes with an unreachable atom ID route, thus rendering the prefixes unreachable.

We can improve or adapt the basic algorithm in several ways, and provide knobs to be tuned by the local AS. A possible improvement to the basic algorithm is to give preference to atoms originated by the local AS.

```
01.select_atom(atomised_prefix p)
02.begin
03.  eligible_atoms = { atom-id i |
04.    ∃ atom a:  a has atom-id i ∧
05.    a declares p part of a };
06.  reachable_atoms = { atom-id i |
07.    i ∈ eligible_atoms ∧
08.    i is reachable };
09.
10.  if ( |reachable_atoms| ≥ 0 )
11.    return atom a such that:
12.      a has atom-id i ∈ reachable_atoms ∧
13.      ∀ j ∈ reachable_atoms:  i <= j;
14.  else
15.    return atom a such that:
16.      a has atom-id i ∈ eligible_atoms ∧
17.      ∀ j ∈ eligible_atoms:  i <= j;
18.  end
19.end
```

Fig. 11. Example of edge router conflict resolution algorithm.

[23] We have omitted inbound and outbound policy on BGP routes from the figure. An operator applies policy on atom ID routes in precisely the same way as today.

[24] We would rarely expect a prefix to be declared part of more than two or three atoms at the same time.

[25] By a *reachable prefix* we mean a prefix that has a valid route.

Note that misconfiguration of overlapping atoms is easily detected: after convergence, atoms are not supposed to overlap. A similar type of misconfiguration in BGP, the accidental announcement of a prefix by an AS [35] (possibly leading to unreachability of the prefix), is hard to distinguish from the case where multiple ASes can reach a prefix and therefore intentionally announce it. Of course atom ID routes, which are announced and routed through BGP, remain as vulnerable to misconfiguration as prefix routes are today.

After applying conflict resolution, the edge router associates each atomised prefix with exactly one atom ID. We use the output of conflict resolution in the remainder of the Atoms Decision Process (Figure 10), which we now discuss.

B.2 Maintaining the Encapsulation Table

The contents of the encapsulation table determines what IP packets get encapsulated and how, and is maintained by the Atoms Decision Process. The Atoms Decision Process places an atomised prefix, together with the atom ID of the containing atom, in the encapsulation table, provided the atom ID is reachable. The Decision Process removes the atomised prefix if the atom ID becomes unreachable, or if the prefix leaves the atom. However, an edge router never makes encapsulation table entries for atoms issued by an atom originator in the same AS, i.e. for membership table entries whose *origin AS* field (Figure 7) matches the AS of the edge router. We return to this issue in Section X-D.

B.3 Generating Atomised Prefix Routes

Similarly, when an atom ID becomes reachable, the Atoms Decision Process generates BGP announcements for the atomised prefixes in the atom to routers outside the DFZ. When an atom ID becomes unreachable, the Atoms Decision Process generates withdrawals for the atomised prefixes. Since we base the attributes of a generated atomised prefix route on those of the atom ID route (Section VI), changes to a reachable atom ID's attributes cause the Atoms Decision Process to reannounce the atomised prefixes with modified attributes. In addition, the Atoms Decision Process sends announcements and withdrawals in response to updates to the membership of an atom which has a reachable atom ID.

C. Garbage collection

The atom membership protocol currently does not have withdrawal messages nor does it withdraw atoms due to membership session disconnects. If an atom originator wishes to remove all atomised prefixes from an atom, it must redeclare the atom with an empty prefix set (and a fresh timestamp). However, such an empty atom still occupies a table entry in edge routers. We are therefore considering adding a mechanism for explicitly withdrawing atom declarations.

An explicit withdrawal mechanism does not eliminate unused table entries due to misconfiguration and implementation errors. An additional garbage collection mechanism is therefore needed to ensure that unused entries in edge router tables are eventually cleaned up. A straightforward garbage collection mechanism allows each edge router to keep track of the length of time that has elapsed since the last update to a membership table entry. If this period is longer than a well-defined expiry period ($expiry$ seconds), the edge router removes the table entry. An atom originator is responsible for redeclaring its atoms with a fresh timestamp, every $refresh < expiry$ seconds. Note that the presence of unused table entries (i.e. entries for atoms whose atom IDs are no longer reachable), and thus the value of these timers, does not affect reachability: as we have seen, an edge router effectively ignores the presence of atomised prefixes in atoms whose atom IDs are unreachable when the edge router generates routes, maintains the encapsulation table, and resolves conflicts between overlapping atoms. High $expiry$ and $refresh$ values (e.g. on the order of weeks) reduce the number of update messages due to refreshes, but may increase the number of unused table entries in edge routers. The exact mechanisms for garbage collection and what timer values are appropriate require further investigation.

IX. Convergence Properties

We will determine quantitive convergence properties of the atomised routing architecture through analysis and simulation (Section XVI). In this section we speculate about some of the expected advantages and disadvantages in terms of convergence behaviour. We divide convergence properties of our architecture into three categories: atom membership, BGP, and the combination of the two.

A. Atom Membership Protocol

The atom membership protocol is a new protocol whose convergence properties we can measure by simulation, but cannot yet determine in an operational setting. However, we designed the protocol with known convergence problems of the current interdomain routing system in mind. In Section VII we mentioned that a session reset in the membership protocol does not have so widespread an effect as a BGP session reset. Below we discuss other differences between the atom membership protocol and BGP, and review remaining issues particular to the membership protocol.

A.1 Delayed Convergence

The atom membership protocol does not suffer from delayed convergence behaviour in the manner that BGP does. Delayed convergence in BGP is generally considered to be caused by BGP path exploration [33]. In contrast, an atom membership update does not contain an AS path, nor does propagation of a membership update depend on any other preference of one membership update over another, based on the path taken by updates. Therefore, the specific problem of delayed convergence due to path exploration does not occur in the membership protocol. However, until we have carried out simulations, we cannot confidently rule out other potential convergence problems.

A.2 Network Size

Since only a subset of interdomain routers participate in the atom membership protocol, the membership protocol operates at a smaller scale than BGP. The network of membership speakers has a smaller diameter than the BGP network, and hence we expect updates to exhibit lower latency due to processing by fewer

routers. Also, the maximum number of routers that receive updates following a change is smaller for a membership change than for a BGP routing change. On the other hand, a BGP routing change does not necessarily update all BGP routers, even if the routing change concerns a globally routed prefix. For example, consider a BGP router that has received two routes for a prefix, r_1 and r_2, and prefers r_1. If the router subsequently receives an update for r_2 and still prefers r_1, then the router will not propagate the update to other routers. In contrast, in the membership protocol, a membership change updates all edge routers unconditionally.

A.3 Accounting for Lost Packets

Another interesting difference between BGP and the atom membership protocol, with regard to convergence behaviour, is what happens to IP packets that are in transit while convergence takes place. Consider an AS A that withdraws prefix p in BGP, and assume that no other AS announces p. After the withdrawal event has taken place, BGP typically explores numerous paths [33] (all errant), before converging. During convergence, a packet destined for p will follow a path but will eventually be dropped by some router r. If router r is outside the AS that withdrew p, the packet is not charged [28] to A. In effect, r's AS R pays for a packet that is dropped as a result of the withdrawal initiated by A.[26] In contrast, in the case of convergence of the equivalent atom membership protocol event (i.e. an atom originator removes prefix p from an atom), IP packets destined for p stand a better chance of being delivered to A or dropped early. When such an IP packet encounters the first edge router, there are two cases: either the edge router has learned of the membership event that is converging, or it has not. If it has, the edge router drops the packet, as though convergence of the event had completed. If the edge router has not learned of the event, the edge router encapsulates and forwards the packet as though the event had not yet occurred. In short, the packet is either dropped early at the edge router, incurring little cost, or else it is encapsulated, forwarded, and delivered to A without encountering the effects of membership convergence behaviour resulting from the withdrawal. In the latter case, AS A pays for the delivery of the packet.

A.4 Forwarding and Signaling Path

After an IP packet has passed through an edge router and is encapsulated, its forwarding path is governed by BGP, independent of the signaling path of the atom membership protocol. The authors of [23] warn of anomalies such as forwarding loops that may arise if the relationship between forwarding paths and signaling paths is unconstrained. Although the authors discuss IBGP rather than interdomain routing, their warning applies to both areas. In our architecture, BGP and the atom membership protocol are responsible for signaling routing changes. In particular there is no clear relationship between the signaling path of the membership protocol and the forwarding path of an IP packet. [23] mentions tunneling as one possible measure to counter anomalies. Indeed, in our architecture we make use of a form of tunneling (encapsulation) to avoid anomalies, in the following away. The first edge router that an IP packet traverses uses its encapsulation table to encapsulate (tunnel) the packet. After this edge router has encapsulated the packet, none of the subsequent edge routers on the packet's forwarding path consult their encapsulation tables, except to determine that they do not need to encapsulate the packet.[27] Instead, every router on the packet's forwarding path consults its FIB when forwarding the packet. Thus, for encapsulated packets the requirement of constraining the relationship between the forwarding path and the signaling path in our architecture reduces to a requirement on BGP alone. This is the same requirement that today's interdomain routing architecture places on BGP.[28]

A related issue is that our architecture permits each edge router to apply its own algorithm in order to resolve overlapping atoms (Section VIII-B.1). Again, tunneling (encapsulation) prevents potential anomalies arising from inconsistencies between the algorithms employed by different edge routers.

A.5 Clustering Related Updates

A prefix p shifting from atom a to atom b requires two updates, one for each atom. Updating the atoms independently may cause a temporary unreachability of p (if edge routers update a before b), a temporary overlap of the atoms (if edge routers update b before a), or an unpredictable combination of the two (if the update message for one atom overtakes the update message for the other).[29] In the case that both atoms belong to the same AS and are managed by the same atom originator, the atom originator may combine such updates together in one membership message, as shown in Figure 7. An edge router that receives a message with multiple updates must propagate the updates in a single message to other edge routers. In this way, updates that belong together propagate together to all edge routers. Furthermore, each edge router should process all updates contained in a message in its Atoms Decision Process before updating its encapsulation table, and before generating routes outside the DFZ.

B. BGP

Although our architecture is in part based on BGP, BGP plays a smaller part in the atomised architecture than in today's interdomain routing system. In particular, the membership protocol, rather than BGP, handles a subset of updates: those governing changes in the $atom\ ID \leftrightarrow prefix\ set$ mapping. Therefore, although typical BGP behaviour such as delayed convergence due to path exploration [33] remains present in our architecture, we

[26] To be more precise, the sender of the packet pays for the first half of the packet's itinerary, $i1$, which is the portion of the itinerary that traverses the sender's providers. The second half of the itinerary, $i2$, which traverses A's providers, should be paid for by A when A receives the packet. However if router r in AS R drops the packet and is on $i2$, AS R pays for the work performed by ASes on $i2$ that the packet has traversed, and does not recover these costs from A.

[27] Since the destination address of the encapsulated packet is based on an atom ID and since an atom ID never appears in an edge router's encapsulation table in the atomised prefix position, every edge router on the encapsulated packet's forwarding path determines that the packet need not be encapsulated. The result of this determination is stable always, and in particular during convergence.

[28] We do not claim that BGP fulfils this requirement, nor do we attempt to solve routing anomalies in the BGP protocol.

[29] We have not excluded the possibility that edge routers reorder updates to different atoms.

expect it to have a smaller impact on average convergence time and the average number of messages sent during convergence.

C. BGP and Atom Membership Combined

Since updates in BGP and the atom membership protocol are signaled independently and may propagate through independent paths, we may expect anomalies to occur due to reordering of BGP events with respect to membership protocol events. In particular, an event at an origin AS that simultaneously changes the BGP route of an atom ID (and its attributes) and the membership of the atom, cannot propagate as a unit, and edge routers will perceive the event as two separate events. Depending on the order in which an edge router receives the membership and BGP messages, the edge router may associate atomised prefixes with BGP attributes of an atom ID in a way that was not intended by the originator. This state of affairs lasts only during convergence of the two protocols: after convergence the end result is well-defined.

In addition, although within the DFZ the membership protocol does not interact with BGP at all, the dynamics in the membership protocol may affect BGP routers outside the DFZ, since edge routers generate BGP routes for atomised prefixes, in part based on the membership state of atoms (Section VI and Figure 5). However, an edge router only generates a BGP route for an atomised prefix toward those BGP routers to which the edge router also propagates the corresponding atom ID's BGP route. Therefore for each atom the influence that the membership protocol has on BGP extends only to areas outside the DFZ in which BGP propagates the atom ID's BGP route.

Finally, we note that the convergence time of events that affect both BGP and the membership protocol is the maximum of the BGP and membership protocol convergence times of those events. This applies to control and data plane [24] convergence times.

D. MRAI

BGP defines a rate limiting *MinRouteAdvertisementInterval* (MRAI) timer [45], and requires that a BGP router not advertise a prefix twice within the MRAI to the same BGP peer. To ensure fast convergence of an unreachability event, the specification requires that a BGP router does not apply MRAI rate limiting to withdrawal messages. (However, some BGP implementations do apply the MRAI to withdrawals.) Apart from rate limiting, studies have shown that an MRAI for advertisements significantly improves convergence behaviour of BGP, both in terms of duration and number of updates [22]. As discussed, the atom membership protocol does not suffer from BGP's withdrawal convergence delay due to path exploration. However, it may be desirable to make some form of rate limiter part of the membership protocol.

Note that if we introduce an MRAI into the architecture, we must take care to preserve clustering of related updates (Section IX-A.5). Another issue is how to propagate an unreachability event quickly. The membership protocol equivalent of a BGP unreachability event is to redeclare an atom without the unreachable prefix.[30] However, a membership router cannot easily distinguish this case from the case that a prefix moves from one atom to another.

Similarly, some membership flap dampening technique similar to BGP route flap dampening [51] may be useful. Since the efficacy of BGP route flap dampening remains questionable [36], we leave this as future work.

X. Atom Origination

So far, this report has described what happens after an atom originator announces and declares an atom, but has ignored the process of making announcements and declarations. We address atom origination in this section, and discuss related issues of decapsulation, and consistency of the atomised architecture with the IGP in the originating AS. Note that many of the procedures and configuration language fragments in these sections are implementation-dependent. They serve as examples only.

A. Decapsulation

As detailed in Section VIII-A, an edge router encapsulates traffic addressed to an atomised prefix and places an IP address based on an atom ID in the destination field of each IP packet. Routers then forward the encapsulated IP packet until the packet reaches the atom originator that announced the atom ID route in BGP (Figure 4). When an atom originator receives the IP packet, it decapsulates it. Decapsulation is a simple operation, and there are few issues. Assuming the atom originator does not expect encapsulated traffic other than packets encapsulated by an edge router, the atom originator can simply remove the outer header of any received encapsulated packet. However, a more conservative approach is for the atom originator to maintain a *decapsulation table*, in which each entry contains an atom ID that the originator is currently advertising. The originator only decapsulates incoming encapsulated traffic whose destination address matches the atom ID of some entry in the table. If there is no such entry, the originator drops the packet. After decapsulation, the atom originator forwards the packet using the original forwarding function. As an optimisation to avoid performing lookups in two tables, the forwarding table and decapsulation table may be integrated into a single table.

B. Centralised Atom Origination

There are two ways that an AS can perform atom origination: centralised (using one atom originator) and decentralised (using several atom originators). We describe centralised origination here, and decentralised origination below.

```
1.  atom declare 4.1.0.0/16 A1
2.  ip prefix-list A1 permit 3.1.0.0/16
3.  ip prefix-list A1 permit 192.2.0.0/16
4.  network 4.1.0.0/16
5.  network 3.1.0.0/16
6.  network 192.2.0.0/16
```

Fig. 12. Configuration of atom A1 in Figure 2 and Table I.

[30] If the unreachable prefix is the only member of the atom, we can alternatively withdraw the atom ID route in BGP.

We start with an example of a configuration for atom A1 of AS A in Figure 2 and Table I. The fragment shown in Figure 12 is part of the BGP configuration file of the atom originator in AS A, and uses an extended Zebra BGP configuration language syntax.

In line 1, we declare an atom, whose atom ID we arbitrarily[31] choose to be 4.1.0.0/16, and whose atomised prefix set is defined by prefix list A1. In lines 2 and 3 we define the prefix list used in line 1. In line 4 we create a route for the atom ID, allowing the atom originator to announce the atom ID in BGP. Similarly, in lines 5 and 6 we create BGP routes for the atomised prefixes. All statements, except *atom declare*, are part of the standard Zebra 0.93b configuration language.

The *atom declare* statement tells the atom originator that it should declare the atom in the membership protocol. The atom originator makes this declaration independently of whether a BGP route exists for the atom ID. The set of atomised prefixes with which the atom is declared in the membership protocol consists of the subset of the prefixes defined by the prefix list that are reachable (have a route), in this case both atomised prefixes. If the set of reachable atomised prefixes changes, the atom originator automatically issues a membership update redeclaring the atom with a fresh timestamp.

The *network* statement for the atom ID (line 4) causes the atom originator to announce a BGP route. Again, this is independent of whether an atom has been declared for the atom ID. Note that we configure creation of a BGP route for an atom ID in the same way as in an unmodified Zebra implementation. Indeed, we may attach BGP attributes (such as communities) to the route e.g. using a *route-map* statement.

The *network* statements in lines 5 and 6 create BGP routes for the atomised prefixes. The atom originator announces these BGP routes in the same way as if the *atom declare* statement were not present, with one exception: it attaches an *atomised marker* attribute to each atomised route (Section VI). Similar to the atom ID's BGP route, configuring a BGP route for an atomised prefix looks the same as in an unmodified Zebra implementation, and allows *route-map* statements, etc., to be applied.

In this example we have manually configured BGP routes for the atomised prefixes. However, it is also possible to import routes for the atomised prefixes from an IGP. In that case lines 5 and 6 are not present. In particular, importing routes from an IGP allows the atom originator to update the atom declaration automatically as a result of IGP (un)reachability of the atomised prefixes.[32]

It is possible for atoms to overlap as a result of misconfiguration by the operator of the AS. The atom originator is responsible for resolving overlapping atoms that it originates before propagating the atom declarations to edge routers. The conflict resolution algorithm for the atom originator resembles, but is not identical to, the edge router algorithm shown in Figure 11. In particular, in contrast with an edge router, an atom originator does not prefer reachable atom IDs to unreachable atom IDs when deciding which of two overlapping atoms should be assigned a common atomised prefix, for the following reason. Since the overlapping atoms are declared by the same administrative domain, we expect the administrator to detect and fix the misconfiguration quickly. There is therefore less danger of an unused (and therefore unreachable) atom accidentally preventing reachability of an atomised prefix that the unused atom shares with some other, reachable atom. Therefore, since preferring reachable atom IDs to unreachable atoms has the side effect of propagating instability of the AS's IGP into the atom membership protocol, an atom originator simply prefers lower atom IDs in its conflict resolution algorithm.

In addition, the atom originator issues updates to BGP and the atom membership protocol in response to configuration and reachability changes. A simple implementation can issue membership updates for each reachability change to one or more atomised prefixes. A more sophisticated implementation may elect to implement some changes by introducing or removing atoms, and shifting prefixes among the atoms. Also, an implementation has a choice between immediately withdrawing the atom ID route of an atom that has become empty, or instead *caching* the route for a limited period of time in case an atomised prefix is subsequently assigned to it. Note that while caching routes for empty atoms may increase interdomain routing stability and improve convergence behaviour, it may also increase membership and BGP table sizes globally.

B.1 Multihomed ASes

For a single-homed AS, one atom originator is sufficient. A small multihomed AS that is internally well-connected (is not likely to partition) can also operate well using a single atom originator, provided inbound traffic does not exceed the atom originator's ability to decapsulate the traffic (Figure 4). If the multihomed AS has several BGP routers, the atom originator announces atom ID routes to the BGP routers through IBGP (Interior BGP) and the IGP. The BGP routers are each configured to announce the atom ID outside the AS or not (and possibly with different BGP attributes), depending on traffic engineering requirements. Under this configuration, an incoming encapsulated IP packet is forwarded based on atom ID routes toward the atom originator, where it is decapsulated and subsequently forwarded based on atomised prefix routes. However, since this solution may take traffic on a detour, it is not suitable for use by a large multihomed AS.

C. Decentralised Atom Origination

For large multihomed ASes that receive a high volume of traffic or have a wide geographical spread, the presence of multiple atom originators, i.e. decentralised atom origination, is essential. As in the case of centralised atom origination, manual configuration determines what prefixes to allow in each atom, what additional attributes (such as communities) to attach to atom ID routes and atomised prefix routes, and what other ASes to announce atom IDs to. However, on the issues of determining what the atomised prefix set of an atom is (based on reachability of the atomised prefixes, as above), and what timestamp to assign to a membership update, it becomes necessary for atom originators to coordinate their actions. A protocol that allows this kind of coordination is beyond the scope of this report.

[31] Of course, 4.1.0.0/16 must be allocated to this AS.

[32] Whether it is desirable to propagate IGP reachability status outside the AS is debatable.

Furthermore, decentralised atom origination requires a solution for a partitioning of the AS in which atom originators become disconnected. A simple solution is as follows. In each partition, the atom originators coordinate to declare and announce new atoms containing the prefixes that are still reachable there, and issue membership updates that shift prefixes out of the original atoms and into the new atoms. To avoid different partitions independently picking different timestamps when they empty the original atoms, the atom originators of an AS should agree on the timestamps to use before the partitioning event occurs. The BGP routes for the original atoms need not necessarily be withdrawn: they can be left intact and reused once the partition heals. Once the partition heals, the atom originators of the AS are able to coordinate how to shift the atomised prefixes back to the original atoms. If the original atoms had been withdrawn, they are announced again. The BGP routes for the new atoms are withdrawn. The impact of this solution on control and data plane convergence times and on downtime [24] requires further investigation, but is beyond the scope of this report.

D. Edge Router and Atom Origination

The final issue we discuss in relation to the AS of the atom originator is the presence of an edge router in the origin AS. In the atoms architecture, we 'protect' IP packets against routing anomalies by tunneling them to an atom originator in the origin AS of the atom (Section IX). After the atom originator decapsulates the packet, the original destination in the IP packet is exposed, and the packet is forwarded based on IGP routes for the atomised prefix toward the network in which the destination host resides. However, the packet is no longer protected against routing anomalies, and a forwarding loop may result. In particular, if an edge router is present in the origin AS and is on the IGP path between the atom originator and the destination host, the edge router might reencapsulate the packet, and subsequently forward it back to the atom originator based on atom ID routes. We prevent this forwarding loop by requiring that an edge router not make entries in the encapsulation table for atoms declared by the local AS (Section VIII-B.2).

XI. Incremental Deployment of Atom-Based Routing

The atoms architecture discussed so far assumes an Internet in which the DFZ is a contiguous 'atomised zone', and every prefix is either an atom ID, an atomised prefix, or a local prefix (Figure 3). In this section, we weaken those assumptions and discuss several modifications to the architecture that allow incremental (or partial) deployment.

A. Atomised DFZ Islands

Here we discuss an incremental or partial formation of the atomised DFZ. We allow a contiguous subset of ASes of the DFZ, called an *(atomised DFZ) island*, to perform the functionality of the DFZ. An island can be as small as one AS, and gradually grow to encompass the full DFZ. We treat DFZ ASes that are not part of the island as non-DFZ ASes.

There is nothing in the architecture that prevents the creation of multiple islands. For example, in Section V we allow IP packets to enter a fully atomised DFZ several times. Therefore even if a DFZ AS does not wish to take part in atomised routing and thereby prevents formation of a contiguous island by other ASes, we can create multiple islands without that AS. As illustrated in Figure 13, one or more edge routers in each island create membership sessions with each other. An island generates atomised prefix BGP routes into the non-atomised DFZ, and drops atomised prefix BGP routes on receiving them from the non-atomised DFZ, using the mechanisms decribed in Section VI. Figure 14 illustrates forwarding.

As the islands grow to include more ASes, some islands can merge into a smaller number of islands. The larger an island becomes, the more benefit it gains from the atomised routing architecture, since larger islands generally contain a smaller proportion of edge routers, which are the most demanding routers

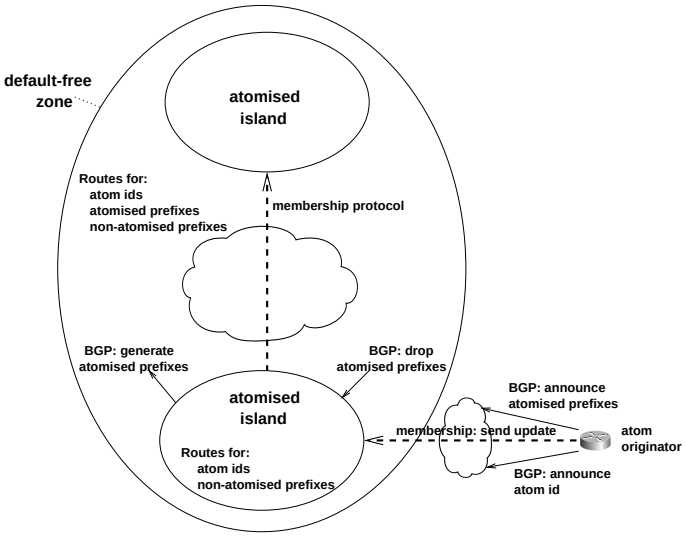

Fig. 13. Incremental deployment: membership and routing.

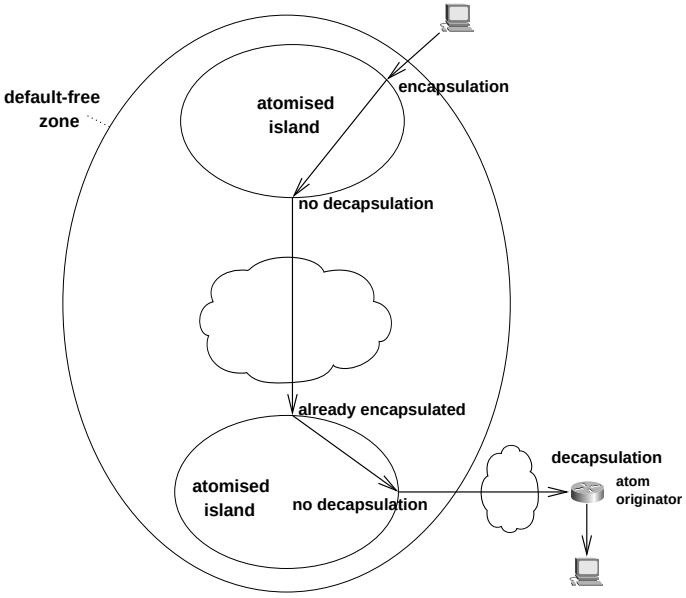

Fig. 14. Incremental deployment: forwarding.

in our architecture. There is therefore an incentive for a DFZ AS to join an existing island, both on the part of the joining AS and the ASes in the island. There is a similar incentive for islands to merge.

B. Non-Atomised Prefixes

A requirement for incremental deployment is the ability to originate and route prefixes without making them part of an atom. In Figure 13 we have shown *non-atomised prefixes* which may be routed as prefixes are today, both inside and outside atomised islands. A special case of non-atomised prefixes are the local prefixes that appear in Figure 3.

There are reasons other than incremental deployment why the atomised routing atchitecture must support non-atomised prefixes. An estimated 49% of declared atoms in the *5 day* dataset consist of a single prefix (53% in the *8 hour* dataset). There is less benefit in creating single-prefix atoms and it would be beneficial to avoid the overhead of doing so.[33] Additionally, a small percentage (around 0.93%)[34] of prefixes in the global routing table are *multi-origin prefixes* (prefixes that are originated by multiple origin ASes) that we cannot easily declare as atoms. We leave these non-atomised. Note that a significant portion of these multi-origin prefixes are likely to have been announced unintentionally: [54] estimates that around 36% of MOAS cases are unintentional, based on their short duration. Finally, we have explicitly set the scope of the atomised routing architecture to target those prefixes that are routed throughout the DFZ today and are not, e.g., aggregated away (Section IV). The remaining prefixes (i.e. local prefixes) can remain non-atomised.

In our architecture, all routers remain BGP-capable and are able to originate and route non-atomised prefixes. However, routing non-atomised prefixes creates a potential conflict: an edge router may find that a prefix is both atomised (declared part of an atom) *and* non-atomised (has a BGP route without atomised marker attribute) in different updates. This state of affairs can only occur as a result of misconfiguration, or during convergence of a rare event in which a prefix legitimately changes its status from atomised to non-atomised, or vice-versa. We can therefore easily detect misconfiguration.

To resolve this conflict, we slightly modify the Atoms Decision Process (Figure 11). When an edge router finds that a prefix is declared part of an atom and also has a valid BGP route, it should consider the prefix to be non-atomised, and not perform local membership processing for it (i.e. should not enter the prefix into the encapsulation table, nor generate an atomised prefix route for it based on the atom ID attributes). However, like the conflict resolution in Section VIII-B.1, this is a local decision and an edge router may choose to consider the prefix atomised, ignoring the conflicting BGP route, and perform local membership processing for it. Encapsulation during atomised forwarding prevents forwarding anomalies in the case that edge routers resolve such a conflict differently (Section IX-A.4). Whichever view of a prefix an edge router takes, it propagates received membership updates to other edge routers precisely as described in Section VII.

XII. Security

BGP is currently not a secure protocol [38]. BGP routers generally trust BGP updates they receive from their peers, and it is possible for an attacker to masquerade as a legitimate BGP peer. An unsecured atom membership protocol shares a number of security weaknesses of BGP, but also suffers from the following additional weaknesses:

- A successful attack on an atom (whether on the BGP atom ID route or the atom membership information) affects all atomised prefixes contained in the atom, rather than a single prefix.
- BGP propagates routes on a hop-by-hop basis, where each BGP router along the propagation path may decide to filter a route or prefer an alternative route for the same prefix. When it does, it does not propagate the route further. This raises a natural impediment against the spread of an attack, and may limit the scope of an attack to a part of the Internet. In contrast, the atom membership protocol requires edge routers to propagate membership updates globally, and without security measures, an attack can spread globally relatively easily. However at the same time, it is harder to launch an attack against an atom without detection.
- BGP routers prefer more specific prefixes to less specific prefixes during forwarding. If an AS advertises some address space through a prefix p_1, an attack which successfully injects a more specific prefix p_2 into the global routing system changes forwarding behaviour for IP packets destined for an address in p_2. However, since p_2 is more specific than p_1, the attack affects a smaller piece of address space than covered by p_1. In general, the more specific p_2, the more effective p_2 is in overriding overlapping routes,[35] but the smaller the address space it is able to affect.

We have not comprehensively analysed security issues of the atom membership protocol. However, we argue that we can apply a number of BGP security measures to the membership protocol. The atom membership infrastructure resembles the BGP infrastructure in that it is based on routers that peer with one another through TCP-based sessions. As a consequence, many security measures that protect BGP against outsider attacks, such as the TCP MD5 signature option [25], and deployment of IPSEC [31], are equally applicable to the atom membership protocol. However, GTSM [20], another security measure in this category, is less effective at protecting BGP multihop sessions than single-hop sessions. Since the atom membership protocol is a multihop session based protocol, GTSM is not so effective at protecting the atom membership protocol.

Other security measures are able to protect BGP against attacks through legitimate BGP peers, notably SBGP [32] and SoBGP [39]. In particular, these approaches attempt to verify whether the origin AS of a BGP update is authorised to advertise a certain prefix. SBGP additionally provides the ability to verify the AS path of a BGP update. We expect to be able to use modified versions of SBGP and SoBGP to verify whether an AS is authorised to issue atom membership updates for a particular atom ID or containing a particular prefix. Note that there is

[33] There is *some* potential benefit in the sense that a single-prefix atom can take advantage of the convergence behaviour of the membership protocol to signal unreachability etc.

[34] *8 hour* dataset, both start and end snapshots.

[35] Until p_2 exceeds length /24 in which case it is likely to be filtered by many BGP routers.

no need to verify the path through which a membership update travels as there is for BGP.

Additionally, we are considering adding an (unsecured) AS path to atom membership update messages, to better enable auditing and generally aid in debugging. An edge router would not examine such an AS path either when it propagates a membership update nor when it processes a membership update locally (Section VII). Therefore an added AS path does not change the convergence properties of the atom membership protocol described in Section IX.

Another attack against the atom membership protocol is to prevent propagation of atom membership updates. Two specific examples of such an attack against an AS A are (a) to prevent propagation of membership updates issued by an atom originator in AS A to the rest of the Internet, and (b) to prevent propagation of membership updates issued by other ASes in the Internet from reaching AS A. Either attack against AS A requires the attacker to block membership updates on paths through all providers, AS peers, and customers of A.[36] As the Internet becomes increasingly interconnected [26], such attacks become harder to carry out successfully.

Although BGP and the atom membership protocol share this vulnerability, BGP appears to be more vulnerable. BGP relies on multiple instances of updates that arrive at a BGP router through distinct AS paths. In BGP, each such AS path represents a distinct path that the BGP router might send traffic through (subject to policy decisions and AS loop detection), and therefore potentially improves connectivity in the data plane. In contrast, in the atom membership protocol it does not matter how many different paths an edge router receives a particular membership update through, as long as the edge router receives at least one instance of the update. Although the latency of an update message may be affected if the update is prevented from traveling along a low-latency path, that in itself does not affect the data plane.

XIII. Tunneling Issues

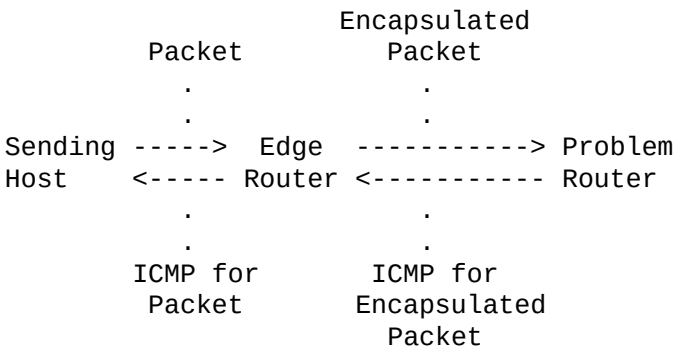

Fig. 15. Relaying ICMP messages.

In our architecture an edge router encapsulates IP packets and tunnels them to atom originators. In this section we review a number of issues regarding tunneling.

The first issue is that any tunneling technology adds a number of bytes to the encapsulated packets, thus reducing the bandwidth available for payload. In order to minimise this effect we specify Minimal IP-in-IP [42] as the encapsulation protocol since it requires a relatively small number of additional bytes (12).

Another issue is the potential performance penalty for adding an encapsulation header during forwarding. However, after discussions with a router vendor we believe that this is not a real issue, since modern routers are generally capable of adding standard encapsulation headers (e.g. MPLS and Ethernet), and in fact are typically optimised to do so.

Part of the specification of IP-in-IP [41] which also applies to Minimal IP-in-IP concerns the behaviour of the encapsulator (an edge router in our case) when it receives an ICMP message that was generated from within the tunnel, as a result of a problem encountered by a packet in the tunnel. For a number of ICMP messages, the edge router is responsible for relaying ICMP messages to the host that sent the unencapsulated IP packet (Figure 15). ICMP requires IP routers to return the IP header of the packet that caused the problem, and at least 8 additional bytes of data beyond the header. To correctly relay an ICMP message, an edge router must remove the IP header (i.e. the encapsulation header) included in the received ICMP message, reconstruct the original IP header, and include at least 8 bytes of additional data. However, since Minimal IP-in-IP inserts a *forwarding header* of 12 bytes after the IP header when encapsulating, we can only guarantee that an edge router recovers the encapsulation IP header and the initial 8 bytes of the forwarding header from the ICMP message. This is insufficient to reconstruct the original IP header[37] let alone to return an additional 8 bytes of data.

To solve this problem, [41] recommends that the encapsulator maintain soft state about the tunnel, based on ICMP messages that it receives from within the tunnel. Before encapsulating an IP packet, the encapsulator checks the tunnel state. If the tunnel state indicates that the encapsulated packet will encounter a problem in the tunnel and trigger an ICMP message, the encapsulator can drop the packet and send an ICMP message describing the expected problem to the sending host. Unfortunately this solution does not scale well in the atoms architecture. The proposed soft state comprises, for each tunnel (i.e. each atom ID), at least the following information [41]:

- MTU of the tunnel
- TTL (path length) of the tunnel
- Reachability of the end of the tunnel

This data significantly increases the state the edge router must maintain on the line card. We are considering ways to reduce the amount of soft tunnel state. For example it may be feasible to use a common, default MTU and only maintain soft state for atom IDs whose MTU is smaller than the default. However, this approach brings a trade-off: by lowering the default MTU we reduce the amount of soft state that an edge router maintains, but at a cost of forcing hosts to send smaller packets than necessary in an increased number of cases.

[36] For these two examples of attacks, blocking updates on paths through the providers of AS A may be sufficient for preventing propagation of updates between AS A and most of the Internet. However, to prevent propagation of updates between AS A and *all* other ASes, the attacker would additionally need to block updates on all paths through AS peers and customers of A.

[37] The original source IP address is missing.

XIV. Granularity of Declared Atoms

We based our estimates of the number of declared atoms (Section III-C) on the assumption that an origin AS partitions its prefixes according to the set of adjacent ASes (which we defined as the *actual origin link set*) to which it wishes to announce each prefix. Since in our architecture routers forward IP packets through the DFZ along a path established by an atom ID route and the atom ID's BGP attributes, atomised prefixes effectively inherit the BGP attributes of the atom ID. Therefore if an origin AS wishes to associate two prefixes with different BGP attributes, it will place them in different declared atoms, independent of whether they are announced to the same set of ASes.

Currently, origin ASes have few means of specifying the policy of a route to non-adjacent ASes. For example, most if not all BGP communities that were part of a recent survey [4] take effect in the AS that attaches the community to a route, or in its adjacent ASes. Also, we have already noted that most policy differentiation of prefixes is observed between the origin AS and its adjacent ASes [1], and that AS path prepending does not refine the number of computed atoms by more than 1%[38] (Section III-A). Therefore most traffic engineering mechanisms today have local scope. However, there are Internet drafts that propose to allow 'action at a distance' through the use of *flexible communities* [34] [2].

In our architecture, atom ID routes and atomised prefix routes have different scope. Atom IDs are globally routed, whereas atomised prefixes are dropped as they enter the DFZ. Thus, an atom originator may specify *global* policy through an atom ID's BGP route, and a more granular, *local* policy through an atomised prefix's BGP route. If indeed origin ASes are interested in refining local rather than global policy, then a distinction between local and global policy in the architecture allows them to do so without affecting the number or stability of globally routed atoms. However, to arrive at a complete distinction between local and global policy, we must solve several problems:

1. Since edge routers prevent BGP routes for atomised prefixes from entering the DFZ, the scope of local policy is determined by how close the origin AS is to the DFZ. In particular, the scope of local policy extends only to adjacent ASes outside the DFZ, and does not include any ASes inside the DFZ.[39] A modification to the architecture that (a) allows BGP routes for atomised prefixes to be dropped, not at the edge of the DFZ, but a certain number of AS hops away from the origin AS (e.g. using BGP communities), and (b) prevents traffic within the scope of local policy from being encapsulated, would make a distinction between local and global policy more effective.
2. In our architecture, an IP packet destined for a particular atomised prefix is encapsulated and forwarded along an atom ID route until it reaches the atom originator. Therefore an attempt to specify local policy for the atomised prefix that is different from that of the atom ID will fail for traffic originated 'globally'. Instead, local policy is only able to affect traffic that originates locally and is not encapsulated. We are considering modifying the architecture to accommodate the ability for routers other than the atom originator to decapsulate traffic. After decapsulation, local policy would govern forwarding of traffic. However, such a modification would have to ensure that the routing anomalies we prevent by means of encapsulation (as described in Section IX) could not occur.

[38] We do not yet have corresponding data for declared atoms.
[39] Policy for the local AS can be implemented in the IGP or MPLS.

XV. Implementation

We have implemented a working prototype of our routing architecture as modifications to the Zebra 0.93b code base [55]. Specifically, the modifications implement an edge router and an atom originator,[40] and follow the architecture we described in this report, except where noted in this section. To begin with, we made the following simplifications relative to the architecture:

- The architecture is applicable to IPv4 and IPv6. However, the current implementation only supports IPv4.
- We did not implement the intra-AS membership protocol (Section VII). Therefore, we support at most one edge router per AS.
- We have implemented the edge router and atom originator roles as separate routers. Currently an edge router cannot originate atoms.
- The *Origin AS* attribute of the membership protocol (Figure 7) is not present in the implementation, nor do we prevent an edge router from making entries in the encapsulation table for atoms declared by the local AS (Section VIII-B.2 and X-D). Therefore this implementation does not permit us to place an edge router inside the AS of an atom originator.
- Clustering multiple updates per atom membership message is not supported (Section IX-A.5).
- We implemented centralised atom origination only. Therefore, we support at most one atom originator per AS (Section X-B).
- We have not implemented garbage collection nor explicit membership withdrawals (Section VIII-C).
- We implemented IP-in-IP encapsulation rather than Minimal IP-in-IP encapsulation (Section VIII-A).
- We did not implement tiny edge routers (Appendix I).

In our implementation, the BGP Decision Process and the Atoms Decision Process are intertwined. To reduce code complexity, we should separate the two along the lines of Figure 10.

A. Forwarding

We did not heavily optimise our implementation, emphasising portability over performance. Nor did we modify the kernel; functions that technically belong in the kernel (encapsulation and decapsulation) we implemented in user space for easier debugging and better portability, at the cost of some performance. Although our current solution relies on FreeBSD *diverted sockets*, porting it to platforms such as Linux should be relatively straightforward.

To implement encapsulation and decapsulation in user space, we used FreeBSD *diverted sockets*. We capture an IP packet from the kernel forwarding path, process it in the Zebra *bgpd* user space process, and place it back on the kernel forwarding

[40] The remaining routers in our architecture (Figure 3) are unmodified BGP routers.

path. In the *bgpd* process we have access to the encapsulation and decapsulation tables, as well as the BGP portion of the forwarding table.

B. MRAI

We implemented a simple MRAI timer for the membership protocol (Section IX-D) as a modification to Zebra's BGP MRAI timer, mainly for the purpose of simulation. We briefly summarise Zebra's BGP MRAI timer. Zebra's MRAI timer is unjittered and operates on a per-peer basis, rather than on a per-peer, per-prefix basis. Every MRAI seconds, a per-peer timer expires. Zebra does not send an outgoing advertisement immediately, but places it on the per-peer queue until the MRAI timer for that peer expires, at which time it sends the advertisement (and any other pending advertisements). Zebra never queues withdrawal messages, but sends them immediately. In addition, Zebra removes queued advertisements when it queues a subsequent outgoing advertisement, or sends a subsequent outgoing withdrawal (for the same prefix and the same peer).

The provisional (and optional) MRAI timer that we implemented for the atom membership protocol in Zebra is similarly unjittered and operates on a per-peer basis. In our implementation a router queues outgoing membership updates on the per-peer queue pending the next MRAI timer expiry for that peer, at which time the router sends any queued membership updates for the peer. The router removes queued membership updates when it queues a subsequent outgoing membership update for the same atom and the same peer. A router never sends membership updates immediately; it always queues updates.

C. Testing

We thoroughly tested the atomised routing architecture and implementation in small topologies (consisting of about fourteen routers), using the *vimage* tool [56] to create virtual PC routers. During testing we had to decide how to assign IP addresses to edge routers and how to route these addresses. We identified two conflicting requirements, and we describe how to resolve these requirements in Appendix II.

XVI. Simulation

In this section we describe the simulations that we intend to carry out. Part of the simulation is based on analysis, which we present first.

A. Inferring Atom Updates

The analysis in this section is based on the *5 day* dataset (Table II) and transforms the BGP update stream in the dataset into a stream of *inferred* updates as issued at origin ASes. We infer two kinds of updates: (1) membership updates on atoms, and (2) BGP updates on atom IDs. In Section XVI-B, we describe how we use this data in combination with simulation results.

Using our approximation of a declared atom as a set of prefixes sharing an origin link set, we determine the (unique) origin link sets for the initial snapshot as in Section III-C. We associate with each origin link set a *prefix set*, i.e. the set of prefixes that share the origin link set. We then process the BGP update stream of the dataset. As we process each BGP update we adjust the origin link sets and prefix sets in accordance with the BGP update.

As a result we see prefixes entering and leaving prefix sets, prefix sets moving from one origin link set to another, prefix sets splitting and joining, etc. We classify these dynamics in terms of the *observed prefix set updates* below and depicted in Figure 16. In describing the prefix set updates, we use the symbols s_1 and s_2 to denote 'source' and 'target' origin link sets, and $P(s)$ as an abbreviation for 'the prefix set of origin link set s':

- *RRC: regular routing change* — All of $P(s_1)$ moves to a formerly empty $P(s_2)$.
- *RSP: regular split* — A proper subset of $P(s_1)$ moves to a formerly empty $P(s_2)$.
- *RJO: regular join* — All of $P(s_1)$ moves to a formerly non-empty $P(s_2)$.
- *RSH: regular shift* — A proper subset of $P(s_1)$ moves to a formerly non-empty $P(s_2)$.
- *ARC: announcement routing change* — Previously unannounced prefixes enter a formerly empty $P(s_2)$.
- *AMC: announcement membership change* — Previously unannounced prefixes enter a formerly non-empty $P(s_2)$.
- *WRC: withdrawal routing change* — All of $P(s_1)$ is withdrawn.
- *WMC: withdrawal membership change* — A proper subset of $P(s_1)$ is withdrawn.

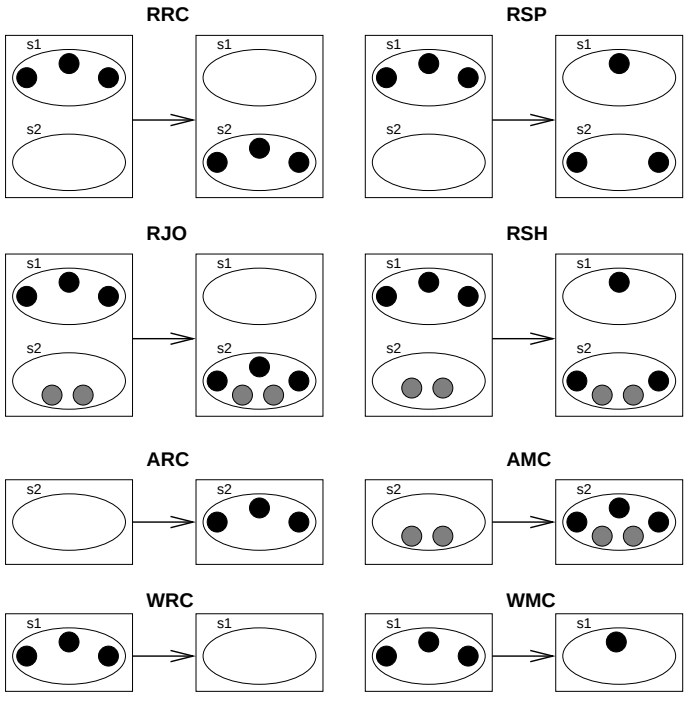

Fig. 16. Prefix set updates.

Having converted the observed BGP updates to observed prefix set updates, we next wish to transform the observed prefix set updates into *actual prefix set updates*. Effectively we attempt to deduce *input signals* to the interdomain routing system by observing its *output signals*. It is unclear to what extent such an analysis can be accurate, since in BGP the relationship between input and output signals is heavily distorted and counterintu-

itive, even for small networks [21]. We identify two types of distortion:

1. The first type of distortion is the convergence behaviour that follows a routing change at an origin AS. For example, Route Views peers likely perceive a single event at its origin AS as occurring at different times. As the event converges, we may observe many different intermediate states that do not correspond to states at the origin AS. Other examples of convergence behaviour are path exploration following a withdrawal of a prefix [33], and route flap dampening suppressing routes for up to an hour even due to relatively simple routing changes [36]. We can counter this type of distortion using a timeout value, as we describe later.
2. The second type of distortion is caused by events in the routing system that are not related to the origin ASes of the affected prefixes, but nevertheless appear as changes in the observed prefix sets associated with origin ASes. As an example of this type of distortion, consider an origin link that the collective Route Views peers observe to be part of the AS path of a single route. If that route is withdrawn due to disrupted connectivity upstream of the origin link, then the origin link may disappear from view completely. We currently do not counter this source of distortion. A more rigorous approach would employ methods such as proposed by [11] to eliminate events that are unrelated to the origin AS.

Note that an origin link may be observable through any number of Route Views peers. We place an origin link in one of the origin link sets if at least one peer sees it, and remove it from its origin link set when we no longer observe it through any peer. As a result, the more peers we use to observe a particular origin link, the more accurate and less noisy the observation should be.

To counter the first type of distortion, we introduce a timeout value t as follows. While applying the BGP update stream, we observe prefixes transitioning from one origin link set to another, as described above. As an example, consider the following two transitions of prefix p: $P(s_1) \rightarrow P(s_2)$ and $P(s_2) \rightarrow P(s_3)$. If p stays with $P(s_2)$ for less than t seconds, we assume that p's association with s_2 is transient, and replace the two transitions with a single transition $P(s_1) \rightarrow P(s_3)$. We repeat this algorithm iteratively (and for every prefix) until we have removed all transient associations between prefixes and origin link sets. After this transformation, we again classify the result into the above prefix set updates.

Note that a small timeout value will be unable to counter the convergence-related distortion. However, a large timeout value will mask many interesting events at the origin AS. It is not clear that a suitable timeout value exists that allows us to observe a significant number of events at the origin AS with reasonable accuracy. Figure 17 shows a breakdown of observed prefix set updates against a varying timeout value. The abbreviations in the plot are as listed earlier, and as depicted in Figure 16. We observe many high-frequency events in the plot. Varying the timeout between 30 and 90 seconds has the greatest impact on the counts of events, which is not surprising if we consider that those Route Views peers that have MRAI timers and implement the MRAI timer on a per-peer basis propagate events at most once every MRAI interval (typically around 30 seconds). We also observe low frequency events that occur on the timescale of hours. These events are unlikely to be artifacts of convergence or route-flap dampening.

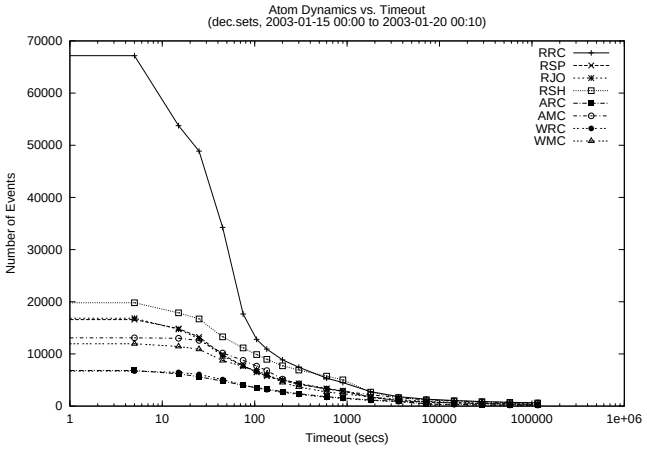

Fig. 17. Breakdown of observed prefix set updates.

Unfortunately the plot does not suggest an appropriate value for the timeout variable. This means that when we use the data to infer *actual prefix set updates*, we should set the timeout value high in order to eliminate as much of the distortion as possible, though at the cost of also removing some actual prefix set updates. In particular, if we choose a timeout value on a human timescale, say 15 minutes to 1 hour, we may be able to capture human-instigated change, such as manual configuration changes, while removing most of the effects of convergence. Ultimately, a study using a multihomed BGP beacon [5] should provide an indication whether the results of the analysis have meaning, and if so, what a good timeout value might be.

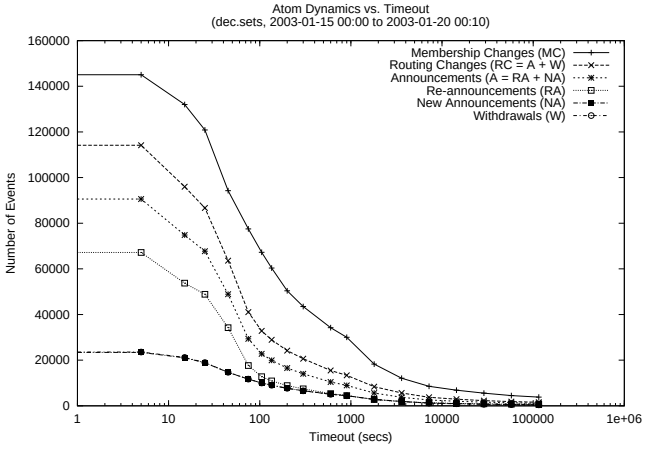

Fig. 18. Breakdown of fundamental updates.

If we consider each prefix set to be a declared atom, we can implement a prefix set update as a combination of BGP routing changes and atom membership changes on the atoms concerned. We break down the updates into *fundamental updates* as in Table VI. For example, an *RRC* could be implemented as a single reannouncement in BGP of the affected prefixes, and an *RSH* could be implemented as two membership updates, one

Prefix set update	BGP updates	Membership updates
RRC	Reannouncement	0
RSP	New Announcement	2
RJO	Withdrawal	2
RSH	none	2
ARC	New Announcement	1
AMC	none	1
WRC	Withdrawal	1
WMC	none	1

TABLE VI
FUNDAMENTAL UPDATES OF EACH PREFIX SET UPDATE.

for each atom.[41] Note that there are several ways to implement a particular prefix set update, and Table VI only lists the most obvious unoptimised implementations. For example, although the most obvious way to implement a *WRC* is to both withdraw the atom ID of s_1 in BGP and declare the atom s_1 empty in the atom membership protocol, this prefix set update may also be implemented by performing only one of these fundamental updates alone. Figure 18 shows the resulting fundamental operations against the timeout value. We observe that the number of atom membership changes appears to be greater than the number of atom ID routing changes. The ratio of membership changes (MC) to routing changes (RC) is approximately 2:1 at larger timescales.

B. Simulation

The simulations that we have planned measure the following properties:

- Convergence behaviour (convergence time and number of updates) of the atom membership protocol. For this we will use simple topologies as in [22]. Specifically we will investigate the cost of structured propagation by the atom membership protocol, and the effect of an MRAI timer.
- Convergence behaviour of the atomised routing architecture. For this simulation we use a larger topology, derived from a subset of the AS graph visible through Route Views. We will perform two types of simulation. First, we will measure the convergence time and number of update messages of each type of fundamental update in Table VI above. By weighting these with the statistics in Figure 18, we will attempt to derive the average number of update messages and average convergence time per update.

In the second simulation, we will convert each of the prefix set updates into equivalent BGP-only updates and again derive average number of update messages and average convergence time per update. In this way, we are able to compare the atomised routing architecture to BGP in terms of convergence properties. We will carry out the simulations using the BGP++ simulator [14] developed by Dimitropoulos and Riley.

XVII. PROVIDER-DECLARED ATOMS

The atoms architecture defines atoms as independent, non-hierarchical sets of atomised prefixes (Section IV). In this section we temporarily depart from this viewpoint and speculate what an architecture would look like if we added an aggregation layer on top of atoms. Specifically, we discuss a layer of *provider-declared atoms*[42] over *origin-declared atoms* as in Figure 19. *Origin-declared atoms* are declared at the origin (owner) AS of a prefix and correspond to the declared atoms we have discussed so far. A *provider-declared atom* consists of the prefixes of a number of *origin-declared atoms* (from different customers) and is declared at and announced from the immediate providers of the origin ASes of the subsumed origin-declared atoms. A provider-declared atom replaces the origin-declared atoms in the global routing system, thus reducing the number of atoms in the global routing system. Based on analysis, the potential savings are quite significant: we estimate a reduction in the number of atoms of up to 50%.

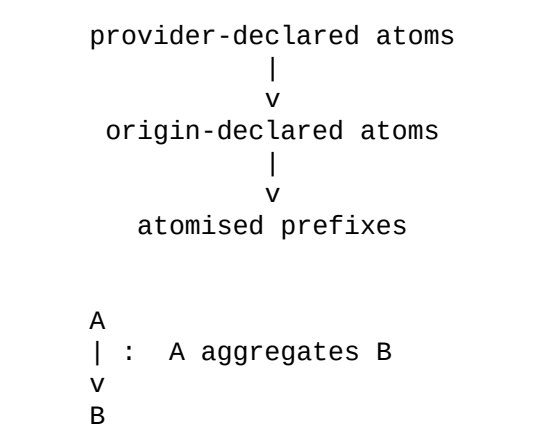

Fig. 19. Adding another aggregation layer.

Provider-declared atoms require a more radical departure from the current interdomain routing architecture than the approach discussed so far. We see the implementation of provider-declared atoms as a possible next step for the atomised routing architecture, one that could be taken after deployment of origin-declared atoms.

A. Example of provider-declared atoms

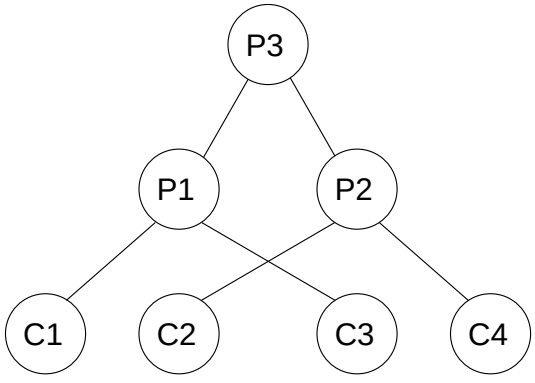

Fig. 20. Provider-declared atoms: example topology.

In Figure 20, Cx are customers, and Px are providers. In addition, P1 and P2 are customers of P3. For the moment, we ignore

[41] *RSH* could be implementated using a single membership *message* if we clustered the updates (Section IX-A.5 and Figure 7).

[42] Provider-declared atoms are a special case of Broido's *crown atoms* [6].

prefixes originated by Px, and examine those originated by Cx. Assume that each Cx makes no distinction among the prefixes it originates, and announces the prefixes to both providers P1 and P2. Then the resulting origin-declared atoms are as listed in Table VII.

Atom	Origin Link Set
O1	{ $P1$ - $C1$ }
O2	{ $P1$ - $C2$, $P2$ - $C2$ }
O3	{ $P1$ - $C3$, $P2$ - $C3$ }
O4	{ $P2$ - $C4$ }

TABLE VII
ORIGIN-DECLARED ATOMS DERIVED FROM FIGURE 20.

As we can see, C2 and C3 announce prefixes to exactly the same set of providers: P1 and P2. As far as reachability to the rest of the Internet is concerned there is no distinction between atoms O2 and O3. Therefore we may as well declare the prefixes in these atoms as a single atom. If the routing system is able to detect cases such as this, and merge such prefixes into a single atom, we can reduce the number of atoms. To allow merging of O2 and O3 to happen, we let immediate providers P1 and P2 of C2 and C3 declare a provider-declared atom subsuming the origin-declared atoms O2 and O3. Similarly, a provider-declared atom subsumes O1 and another provider-declared atom subsumes O4. Table VIII lists the atoms declared by providers. The number of atoms declared is now 3 instead of 4. Note that the example only shows customers that announce their prefixes to all their respective providers. However, it should be clear that providers are similarly able to merge atoms from customers that announce different prefixes to different providers.

Prov. Atom	Subsumes	Declaring Prov.
D1	O1	P1
D2	O2 and O3	P1 and P2
D3	O4	P2

TABLE VIII
PROVIDER-DECLARED ATOMS DERIVED FROM FIGURE 20.

B. Implementation

A customer Cx partitions its prefixes into origin-declared atoms. However, instead of declaring these globally in the membership protocol and announcing atom IDs for them globally in BGP, the customer sends each origin-declared atom to the providers that it wishes the atom to be reachable through, together with a list containing all such providers. The provider is then able to merge the atom with those origin-declared atoms from other customers that share the same provider list, and, as per the atomised routing architecture, declare the resulting provider-declared atom in the membership protocol as well as announce its atom ID in BGP. Note that while a customer is required to reveal to each provider what other providers it has, in today's routing architecture this information is largely available in route collectors such as Route Views.

This scheme requires atom originator routers in different providers (P1 and P2) to communicate with one another in order to implement decentralised atom origination (Section X-C), despite the fact that these providers may not be peering with each other. The most obvious way for the providers to communicate is through the customer networks. Rather than passively forwarding such communication between the providers (which may not be acceptable), routers in the customer ASes may play an active part in the inter-provider communication process, and verify the that the communication between their providers is related to the customer AS's prefixes.

C. BGP Attributes

After merging origin-declared atoms from different customers into a single provider-declared atom, the atomised prefixes concerned share a uniform set of BGP attributes, namely the attributes that are attached to the atom ID route of the provider-declared atom. Yet it is likely that the origin AS announced the origin-declared atoms with distinct attributes. In particular, the ASPath attribute contains a different origin AS for each customer. However, many BGP attributes attached by an origin AS lose their relevance beyond adjacent ASes. For example NextHop, MultiExitDisc, and LocalPref are dropped or replaced [45] before they propagate beyond the adjacent AS. However, other attributes transit multiple ASes, e.g. some extended communities [48]. Providers cannot merge origin-declared atoms unless such attributes are identical.

Atom	ASPath P1	ASPath P2
O2	$(C2)$	$(C2)$
O3	$(C3)$	$(C3)$
D2	$(P1$ - $\{C2, C3\})$	$(P2$ - $\{C2, C3\})$

TABLE IX
MERGING ASPATH ATTRIBUTES.

We treat the ASPath attribute as a special case. BGP provides a rarely used feature called the *ASSet*, which is used to merge the ASPath attributes of different prefixes, or in our case atoms. In Table IX, both providers P1 and P2 see the atom IDs of O2 and O3 with ASPath attribute $(C2)$ and $(C3)$, respectively. When announcing to P3, the normal behaviour is for P1 to extend the ASPaths to $(P1$ - $C2)$ and $(P1$ - $C3)$, respectively, but this prevents merging O2 and O3 into D2. To resolve this conflict, it instead attaches a *merged* ASPath $(P1$ - $\{C2, C3\})$ when announcing D2 to P3. Similarly, P2 announces D2 to P3 with ASPath $(P2$ - $\{C2, C3\})$.

D. Scope

Provider-declared atoms are best applied to prefixes whose origin ASes are *customer-only ASes*, i.e. ASes that do not have customers, e.g. the Cx nodes in Figure 20. While it is reasonable to assume customer-only ASes are willing to delegate global atom declaration and announcement to their providers, providers are likely to want to manage their own atoms. So for

example rather than delegating to P3, P1 and P2 globally declare and announce as (origin-declared) atoms any prefixes they own. This distinction results in a mix of origin-declared and provider-declared atoms, which we refer to as *provider/origin-declared atoms*.

E. Analysis

In this section we analyse the reduction in the number of declared atoms in several steps. First, given a dataset we use the following method to determine what ASes are *stub ASes* and what ASes are *transit ASes*. From the initial snapshot of the dataset we construct the AS graph. We then process the updates of the dataset, modifying the AS graph with each update. As we examine successive AS graphs we add to the set of transit ASes each AS that has an outdegree > 0. After examining all AS graphs, we classify the remaining ASes as stub ASes. In the *5 day* dataset, 12k out of 15k ASes (83%) are stub ASes. We make the simplifying assumption that transit ASes are provider ASes and stub ASes are customer-only ASes.[43]

Second, we divide prefixes in two categories: those originated by transit ASes and those originated by stub ASes. For prefixes originated by transit ASes we count the number of unique origin link sets as before (Section III-C). Of these prefixes we consider those that share the same unique origin link set to be an origin-declared atom, reflecting our assumption that provider ASes do not wish to delegate atom declaration to other ASes.

For prefixes originated by stub ASes, we count the number of unique *provider sets*. A provider set of a prefix p is the set of upstream ends on the origin links of p. For example in Figure 20 the provider set of C2 is $\{P1, P2\}$. Of the prefixes originated by stub ASes, we consider those that share the same unique provider set to be a provider-declared atom, reflecting our assumption that customer-only ASes delegate atom declaration to their immediate providers.

Prefs	C.Atoms	O.Decl.Atoms	O/P-Decl.Atoms	Recurrence
123k	27k	21k	10k	85.6%

TABLE X
ESTIMATED NUMBER OF PROVIDER/ORIGIN-DECLARED ATOMS.

For the *5 day* dataset[44], the estimate of the number of provider/origin-declared atoms that results is 10k (Table X), a 51% reduction compared to the number of origin-declared atoms. However, note that the recurrence ratio of 85.6% is well below that of origin-declared atoms (93.4% in Table IV). In other words, provider/origin-declared atoms appear to be less persistent over a long period of time.

Figures 21 and 22 show the dynamics of provider/origin-declared atoms, analogous to the dynamics of origin-declared atoms in Figures 17 and 18. In general the total volume of dynamics is comparable between origin-declared and provider/origin-declared atoms, though the breakdown by event type is different. In comparison to the statistics on observed prefix set updates for origin-declared atoms (Figure 17), the observed prefix set updates for provider/origin-declared atoms (Figure 21) show half the number of regular routing changes and double the number of regular shifts (RSH). Figure 22 shows a corresponding reduction in the number of BGP routing changes (RC) and increase in the number of membership changes (MC). Thus we see that the proportion of atom membership changes to BGP changes under provider/origin-declared atoms is significantly larger than under origin-declared atoms. We conclude that the performance of the membership protocol relative to the performance of BGP (e.g. in terms of the number of update messages in the routing system that a change at the origin AS incurs) has greater impact in a routing system based on provider/origin-declared atoms than in a routing system based on origin-declared atoms, since provider/origin-declared atoms will involve more membership changes.

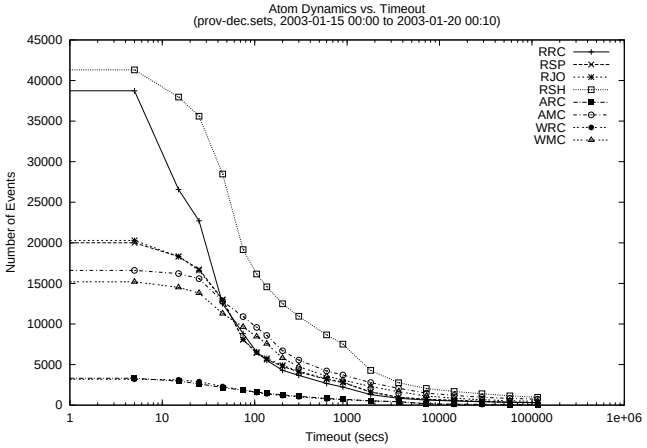

Fig. 21. Observed prefix set updates of provider/origin-declared atoms.

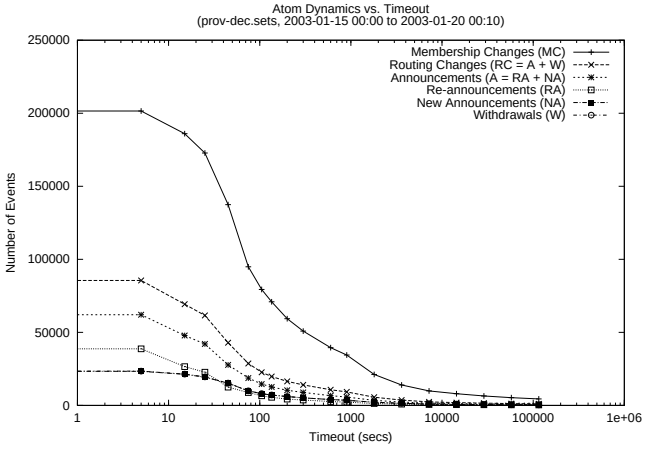

Fig. 22. Fundamental updates of provider/origin-declared atoms.

F. Summary of Provider-Declared Atoms

Compared to origin-declared atoms, provider/origin-declared atoms potentially offer significant savings in the number of declared atoms. Based on Route Views data, provider/origin-declared atoms offers the potential for up to 50% reduction in

[43] In other words, the presence of an AS peering link between two customer-only ASes will make either or both ASes appear to be a provider AS. However, it is unlikely that an AS peering link between two customer-only ASes is visible in Route Views.

[44] We did not compute an estimate for the *8 hour* dataset.

the number of declared atoms. The savings are less if customer-only ASes attach to their atoms distinct BGP attributes that transit multiple ASes, since the providers of these customer-only ASes will not be able to merge such atoms. On the other hand, the current trend of an increasing number of stub ASes compared to transit ASes [9] [27] produces a tendency to increase the savings, since on the average each provider will have a larger number of customer-only ASes whose atoms the provider may merge. These two factors interact, since these new stub ASes may attach distinct attributes to their atoms, so we cannot clearly assess the net savings.

Provider-declared atoms face a number of technical and non-technical hurdles. First, they require stub ASes to divulge possibly sensitive information to and about their providers. Second, they require a degree of cooperation among competing ISPs and a means for providers to communicate without peering with each other.

XVIII. Future Work

Throughout this report we have indicated issues that remain unresolved or unimplemented. We summarise those issues here, referring to relevant sections for details.

Our first priority is to carry out the simulations in Section XVI-B. These validate or discount several assumptions underlying the atomised routing architecture. Next, there are a number of architectural issues that we must address:

- security measures (Section XII).
- scalability of tunneling (Section XIII).
- distinction between local and global policy (Section XIV).
- decentralised atom origination (Section X-C).
- rate limiting and flap dampening (Section IX-D).
- reducing overhead of the atomised routing architecture by using one of an atom's atomised prefixes as the atom ID (Section XIX).
- exploiting provider-declared atoms in a practical way (Section XVII).

Finally, we need to finish our prototype implementation to conform to the architecture that we described in this report, as indicated in Section XV.

XIX. Discussion

The atomised routing architecture offers a novel approach toward aggregation of prefixes beyond what is possible in CIDR, thus reducing the number of globally routed BGP objects and reducing global BGP table size. In addition it is able to perform a subset of routing updates outside of BGP, thus avoiding BGP's convergence problems for those updates. Finally, the atoms architecture offers a way of distinguishing local updates from global updates. However, we must set off these advantages against the following concerns:

- The atomised routing architecture adds the atom membership protocol to the interdomain routing system, thus increasing the complexity of the whole. To deploy a new protocol without decreasing the robustness of the system, it should be trivially configurable (or self-configurable) and secure. We have outlined ways that the protocol could be made somewhat self-configurable at the origin AS (Section X-C), but more work is needed. We can with some success detect misconfigurations of peering sessions (Section VII), and we believe that we can secure the protocol using existing technologies (Section XII).
- Part of the atomised routing architecture is an encapsulation (tunneling) mechanism. It is unclear how well tunneling will serve as a general-purpose mechanism in an Internet with varying MTUs, and routers that may generate ICMP messages containing no more than 8 bytes of additional data beyond an IP header (Section XIII).
- Further aggregation of the IP address space decreases the ability of transit ASes to perform traffic engineering based on the BGP routes of other ASes. Such a trade-off is inevitable in any proposal that attempts to reduce the number of globally routed objects.
- Security of this (any) system brings a similar inevitable trade-off. Further aggregation of IP address space provides larger aggregates as targets of attack. On the other hand, a new protocol has the opportunity to integrate security measures from the start.
- While our atoms architecture could reduce the number of global BGP routes, it makes individual atomised prefixes globally reachable through the atom membership protocol. Thus, we have not removed state from the routing system as a whole, but moved it from BGP into the atom membership protocol.

At the level of routers, we are able to decrease the table size of the transit routers inside the DFZ, and have moved this state into the edge routers of the DFZ. We expect that in a fully deployed setting the edge router role would be played by routers in end customer sites and by some or all access routers in an ISP network. Access routers that peer with customers that are not in the DFZ would become edge routers. Core routers and the subset of access routers that peer only with other DFZ ASes would take on the role of transit routers.[45] An access router generally carries less traffic than a core router, and consists of cheaper, commoditised hardware. Thus it makes sense to assign the edge router role, and its increased memory requirement, to the access router. This configuration allows core routers to take advantage of the decreased memory requirements of the transit router role. However, consider a partially deployed setting (Figure 13) in which a large ISP A in an atomised island peers with a large ISP B that is not part of an atomised island, and assume that the two ISPs exchange large volumes of traffic. At first sight, it appears that the router R that peers on behalf of ISP A must act as a high throughput edge router for traffic it receives from ISP B. We can avoid the introduction of a high throughput edge router under the following assumptions. In a partial deployment, we expect a limited amount of traffic destined for atomised prefixes. Instead of acting as an edge router and encapsulating this traffic, router R may forward the traffic to a lower capacity edge router E for encapsulation (e.g. using a default route), while maintaining high throughput for traffic destined for non-atomised prefixes received from ISP B. As the proportion of atomised prefixes increases, the volume of traffic destined for atomised traffic also increases, which places greater requirements on E. However, we expect an increased number of atomised prefixes to act as an incentive for ISP B to start supporting atomised routing. In general we expect peerings between atomised and non-atomised

[45] Alternatively, the edge router role could be at the distribution router level, a notion used in the provider world for routers responsible for aggregating customer routes before propagating them to the core.

ISPs to eventually be replaced by peerings between atomised ISPs. Once ISP B supports atomised routing, R no longer receives traffic from ISP B that requires encapsulation.

- We increase the number of routes outside the DFZ. The increase is relatively small: we add an atom ID route for each locally originated but globally routed atom. Although the increase is not large, it does affect routers designed with smaller capacity. We can mitigate this effect if we use one of the atomised prefixes in an atom as the atom ID. We leave this as future work.

A. Future Routing Table Growth

Currently, global routing table size is not a major concern in the ISP community. However, there are two pending changes to the Internet architecture that may cause a significant growth in global routing table size. The first is the introduction of IPv6. IPv6 defines a tremendous IP address space, and with it the potential for an AS to splinter prefixes for the purpose of traffic engineering to a greater degree than is possible today. However, IPv6 largely inherits the routing architecture of IPv4 and provides no solution that contains the number of routes that may result.

Another change we expect is the introduction of 32-bit AS numbers [52]. With 32-bit AS numbers deployed, the number of ASes in the routing system may increase dramatically, and with it the number of global routes. We expect such an increase to occur mainly at the edge of the network, i.e. to increase the number of customer-only ASes (Section XVII-D). In that case a solution such as provider-declared atoms could contain the growth in the number of routes due to a large number of multi-homed, customer-only ASes. An IETF working group (*multi6*) is addressing this problem from the multihoming perspective. It remains to be seen whether the group can come up with a solution that will be accepted.

Some members of the network operator community have argued that with the advent of Virtual Private Networks (VPN), the routing table size of an ISP will increase significantly, with or without a global routing table expansion. However, we argue that whereas VPN routes are largely contained within the ISP that offers VPN service (in return for financial compensation), the global routing table exerts a 'pressure' on any DFZ ISP, as well as on smaller customers wishing to carry a default-free routing table for the purpose of improving the quality of their Internet connectivity (Appendix I). In addition, a smaller global routing table size leaves more room for the implementation of services such as VPN.

B. Alternative Approaches

In this section we present alternative approaches we have considered.

Initially, our routing architecture focused on reducing per-prefix processing by a BGP router on a per-BGP-update basis. It did not attempt to remove atomised prefixes from routers (i.e. transit routers in our current architecture), but merely made explicit the grouping of prefixes by common attributes, in the form of atoms. A BGP router that received an update containing an atom would run its BGP Decision Process (DP) and prefix-based policy once for the atom, and apply the results to the atomised prefixes. However, feedback from the vendor community indicated that the overhead in processing a BGP update message lies not so much in the Decision Process but in other, per-update processing. For example, applying policy to BGP attributes appears to be more expensive than applying per-prefix policy. Therefore there is little incentive to optimise per-prefix processing through architectural changes. Note that although BGP has the ability to place multiple prefixes with identical BGP attributes in a single BGP update message, an optimisation to run the BGP DP once over a set of equivalent prefixes remains non-trivial, since the router has no guarantee that the prefixes in an update message are equivalent. Specifically, the router may have other routes for the prefixes that have distinct attributes.

During our discussions at IETF the concept of *null atoms* arose. A null atom is a declared atom that is initially empty, but nevertheless routed. An AS announces a null atom for every possible policy that its prefixes may have, whether currently or in the future (e.g. as a result of a partitioning of the AS, Section X-C). Subsequently the router assigns and reassigns prefixes to these atoms in response to changes to reachability and policy, exclusively through the membership protocol. While this approach avoids BGP updates for such dynamics, it has the disadvantage of increasing the number of BGP routes significantly. For example, a dual-homed AS may announce BGP routes for four different null atoms based on possible reachability conditions alone (i.e. reachable through both providers, neither provider, or either one provider). An increase in the number of BGP routes inevitably leads to an increase in the number of BGP updates that are required to maintain the routes. For example, a loss of connectivity, whether between the origin AS and one of its providers or elsewhere in the network, will lead to a withdrawal of the null atoms advertised on that link, and after repair a reannouncement of the atoms. Therefore it is unclear whether null atoms would ultimately reduce the overall number of BGP updates. However, *assuming* a stable core, one could imagine the concept of null atoms used to bring the Internet closer to sub-second convergence behaviour, provided the membership protocol converges at this speed. Note that we have made use of empty routed atoms in Section X-C where, during a partitioning of an AS, we keep an atom ID route for an empty atom available until the partitioning heals.

Another application of the declared atom concept is Virtual Private Networks (VPN). We believe that the atoms architecture, applied at a smaller scale, can implement VPNs through IP encapsulation, and may serve as an alternative to MPLS-based VPNs such as [46].

XX. Acknowledgements

We would like to acknowledge the following people for providing feedback or helping in other ways: Andrew { Lange, Moore, Partan, Tanenbaum }, Bill Woodcock, Bradley Huffaker, CAIDA folks, Cengiz Alaettinoglu, Daniel Karrenberg, Dave Meyer, Dennis Ferguson, Dino Farinacci, Evi Nemeth, Fontas Dimitropoulos, Frances Brazier, Frank Kastenholz, Geoff Huston, George Riley, Henk Uijterwaal, Jeffrey Haas, Maarten van Steen, Nevil Brownlee, Marko Zec, Mike Lloyd, Pedro Roque Marques, Sean Finn, Senthilkumar Ayyasamy, Ted Lindgreen, Teus Hagen, Vijay Gill, Wytze van der Raay.

REFERENCES

[1] Yehuda Afek, Omar Ben-Shalom, Anat Bremler-Barr, 'On the structure and application of BGP policy Atoms', ACM SIGCOMM Internet Measurement Workshop (IMW), November 2002

[2] S. Agarwal, T. G. Griffin, 'BGP Proxy Community Community', January 2004, INTERNET DRAFT, http://www.ietf.org/internet-drafts/draft-agarwal-bgp-proxy-community-00.txt

[3] T. Bates, R. Chandra, E. Chen, 'BGP Route Reflection, An Alternative to Full Mesh IBGP', RFC 2796, April 2000

[4] O. Bonaventure, B. Quoitin, 'Common utilizations of the BGP community attribute', June 2003, INTERNET DRAFT, http://www.watersprings.org/pub/id/draft-bonaventure-quoitin-bgp-communities-00.txt

[5] Randy Bush, Minutes Routing SIG, August 2003, http://www.apnic.net/meetings/16/programme/transcripts/routing-sig.txt

[6] Andre Broido, kc claffy, 'Analysis of RouteViews BGP data: policy atoms', Proceedings of the Network-Related Data Management workshop, Santa Barbara, May 23, 2001

[7] Andre Broido, kc claffy, 'Complexity of global routing policies', http://www.caida.org/outreach/papers/2001/CGR/

[8] A. Broido, k. claffy, 'Internet topology: connectivity of IP graphs', in SPIE International symposium on Convergence of IT and Communication, Denver, CO, Aug 2001

[9] Andre Broido, Evi Nemeth, kc claffy, 'Internet Expansion, Refinement, and Churn', European Transactions on Telecommunications, January 2002, http://www.caida.org/outreach/papers/2002/EGR/

[10] T. Bu, L. Gao, D. Towsley, 'On characterizing BGP routing table growth', Proc. IEEE Global Telecommunications Conf. (GLOBECOMM), pp. 2197-2201, Nov. 2002

[11] M. Caesar, L. Subramanian, R. Katz, 'Root cause analysis of Internet routing dynamics', U.C. Berkeley Technical Report UCB/CSD-04-1302, November 2003

[12] R. Chandra, P. Traina, T. Li, 'BGP Communities Attribute', RFC 1997, August 1996

[13] D. Chang, R. Govindan, J. Heidemann, 'An Empirical Study of Router Response to Large BGP Routing Table Load', Tech. Rep. ISI-TR-2001-552, USC/Information Sciences Institute, December 2001.

[14] X. Dimitropoulos, G. Riley, 'Creating Realistic BGP Models', Proc. Eleventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03), pp. 64-69, October 2003

[15] S. M. Doran, NANOG mailing list, Mon May 05 12:21:08 2003, 'Internet core scale and market-based address allocation', http://www.merit.edu/mail.archives/nanog/2003-05/msg00123.html

[16] D. Farinacci, T. Li, S. Hanks, D. Meyer, P. Traina, 'Generic Routing Encapsulation (GRE)', RFC 2784, March 2000

[17] V. Fuller, T. Li, J. Yu, K. Varadhan, 'Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy', RFC 1519, September 1993

[18] L. Gao, 'On inferring autonomous system relationships in the Internet', in Proc. IEEE Global Internet Symposium, Nov 2000

[19] L. Gao, J. Rexford, 'Stable Internet routing without global coordination', in Proc. ACM SIGMETRICS, June 2000

[20] V. Gill, J. Heasley, D. Meyer, 'The Generalized TTL Security Mechanism (GTSM)', October 2003, INTERNET DRAFT, http://www.watersprings.org/pub/id/draft-gill-gtsh-04.txt

[21] T. Griffin, 'What is the sound of one route flapping?', slides, July 2002 http://www.cs.dartmouth.edu/~mili/workshop2002/slides/griffin_dartmouth_20020723.pdf

[22] T. Griffin, B. Premore, 'An Experimental Analysis of BGP Convergence Time', In Proceedings of ICNP, November 2001

[23] T. Griffin and G. Wilfong, 'On the Correctness of IBGP Configuration', in Proceedings of ACM SIGCOMM, 2002

[24] F. Hao, P. Koppol, 'An Internet scale simulation setup for BGP', ACM SIGCOMM Computer Communication Review, Volume 33, Number 3, July 2003

[25] A. Heffernan, 'Protection of BGP Sessions via the TCP MD5 Signature Option', RFC 2385, August 1998

[26] G. Huston, 'Analyzing the Internet's BGP Routing Table', The Internet Protocol Journal, Volume 4, Number 1, March 2001, http://www.potaroo.net/papers/ipj/4-1-bgp.pdf

[27] G. Huston, 'BGP Table Statistics', http://bgp.potaroo.net/

[28] G. Huston, 'Interconnection, Peering and Settlements, Part II', in Internet Protocol Journal, June 1999

[29] Y. Hyun, A. Broido, k. claffy, 'Traceroute and BGP AS Path Incongruities', Tech. rep., CAIDA, Mar 2003

[30] IETF multi6 working group charter, 'Site Multihoming in IPv6 (multi6)', http://www.ietf.org/html.charters/multi6-charter.html

[31] S. Kent, R. Atkinson, 'Security Architecture for the Internet Protocol', RFC 2401, November 1998

[32] S. Kent, C. Lynn, J. Mikkelson, K. Seo, 'Secure Border Gateway Protocol (S-BGP)', Proceedings of ISoc Network & Distributed Systems Security Symposium, Internet Society, Reston, VA, February 2000

[33] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, 'Delayed internet routing convergence', in Proc. ACM SIGCOMM '00 (Stockholm, Sweden, 2000), pp. 175–187

[34] A. Lange, 'Flexible BGP Communities', June 2003, INTERNET DRAFT, http://www.ietf.org/internet-drafts/draft-lange-flexible-bgp-communities-01.txt

[35] R. Mahajan, D. Wetherall, and T. Anderson, 'Understanding BGP Misconfiguration', in Proceedings of ACM SIGCOMM 2002

[36] Z. M. Mao, R. Govindan, G. Varghese, R. H. Katz, 'Route Flap Damping Exacerbates Internet Routing Convergence', in Proc. of ACM SIGCOMM 2002, Pittsburgh, PA, Aug 2002

[37] Z. M. Mao, J. Rexford, J. Wang, R. H. Katz, 'Towards an accurate AS-level traceroute tool', Proceedings ACM SIGCOMM Conference, Karlsruhe, Germany, August 2003

[38] S. Murphy, 'BGP Security Vulnerabilities Analysis', June 2003, INTERNET DRAFT, http://www.ietf.org/internet-drafts/draft-ietf-idr-bgp-vuln-00.txt

[39] J. Ng, 'Extensions to BGP to Support Secure Origin BGP (soBGP)', June 2003, INTERNET DRAFT, http://www.watersprings.org/pub/id/draft-ng-sobgp-bgp-extensions-01.txt

[40] W. B. Norton, 'Internet service providers and peering', Proceedings of NANOG 19, Albuquerque, New Mexico, June 2000

[41] C. Perkins, 'IP Encapsulation within IP', RFC 2003, October 1996

[42] C. Perkins, 'Minimal Encapsulation within IP', RFC 2004, October 1996

[43] J. Postel, 'Internet Protocol', RFC 791, September 1981

[44] Y. Rekhter, T. Li, 'An Architecture for IP Address Allocation with CIDR', RFC 1518, September 1993

[45] Y. Rekhter, T. Li, 'A Border Gateway Protocol 4 (BGP-4)', RFC 1771, March 1995

[46] E. C. Rosen, Y. Rekhter, T. Bogovic, S. J. Brannon, M. Carugi, C. J. Chase, T. Wo Chung, E. Dean, J. De Clercq, L. F., P. Hitchen, M. Leelanivas, D. Marshall, L. Martini, M. J. Morrow, R. Vaidyanathan, A. Smith, V. Srinivasan, A. Vedrenne, 'BGP/MPLS IP VPNs', May 2003, INTERNET DRAFT, http://www.watersprings.org/pub/id/draft-ietf-ppvpn-rfc2547bis-04.txt

[47] E. Rosen, A. Viswanathan, R. Callon, 'Multiprotocol Label Switching Architecture', RFC 3031, January 2001

[48] S. R. Sangli, D. Tappan, Y. Rekhter, 'BGP Extended Communities Attribute', INTERNET DRAFT, http://www.ietf.org/internet-drafts/draft-ietf-idr-bgp-ext-communities-06.txt

[49] J. W. Stewart, 'BGP4: Inter-Domain Routing in the Internet', Addison-Wesley, 1999

[50] University of Oregon's RouteViews project, http://www.route-views.org/

[51] C. Villamizar, R. Chandra, R. Govindan, 'BGP Route Flap Damping', RFC 2439, November 1998

[52] Q. Vohra, E. Chen, 'BGP support for four-octet AS number space', February 2004, INTERNET DRAFT, http://www.ietf.org/internet-drafts/draft-ietf-idr-as4bytes-07.txt

[53] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, L. Zhang, 'Observation and Analysis of BGP Behavior Under Stress', in Proc. of ACM SIGCOMM Internet Measurement Workshop 2002, Marseille, France, Nov 2002

[54] X. Zhao, D. Pei, L. Wang, Dan M., A. Mankin, S. F. Wu, L. Zhang, 'Detection of Invalid Routing Announcement in the Internet', in Proc. of International Conference on Dependable Systems & Networks (DSN 2002), June 23-26, 2002

[55] GNU Zebra, Routing software distributed under GNU General Public License, http://www.zebra.org/

[56] M. Zec, 'Implementing a Clonable Network Stack in the FreeBSD Kernel', USENIX Annual Technical Conference, pp. 137-150, 2003

[57] X. Zhao, M. Lad, D. Pei, L. Wang, D. Massey, S. F. Wu, L. Zhang, 'Understanding BGP Behavior Through A Study of DoD Prefixes', in Proceedings of DISCEX III, April 2003

APPENDIX

I. TINY EDGE ROUTER

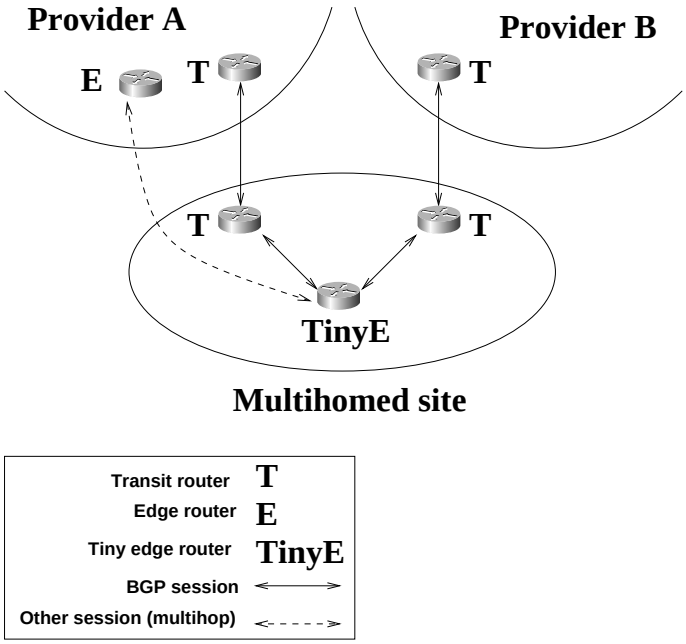

Fig. 23. Multihomed site with tiny edge routers.

We briefly consider the special case of a small multihomed site that wishes to carry a default-free routing table for the purpose of effectively performing outbound traffic engineering, but does not wish to maintain a BGP or membership table containing every atomised prefix. We extend the atomised routing architecture for this case in the following manner. We allow the multihomed site to maintain one or more *tiny edge routers* (Figure 23) that perform the encapsulation function of an edge router, but do not generate atomised prefix routes, nor maintain full membership tables. In addition the site contains transit routers that maintain BGP peering sessions with transit routers in the provider networks. Through these BGP peering sessions the site receives global atom ID routes. Recall that transit routers do not carry routes for atomised prefixes (Figure 3).

Rather than maintaining full membership sessions with other edge routers, a tiny edge router *requests* membership information from other edge routers (dashed line in Figure 23). Specifically, when a host in the site wishes to send an IP packet to a host outside the site, the routers in the site forward the packet to a tiny edge router (e.g. using default routes). The tiny edge router then requests an edge router in one of the site's providers to look up the destination address of the packet in the membership table, and to return the corresponding atom ID. The tiny edge router encapsulates the IP packet using the received atom ID. After the router has encapsulated the packet, the tiny edge router and the default-free transit routers of the site are able to forward the packet out of the site, applying outbound traffic engineering based on the full complement of atom ID BGP routes. As an optimisation, the tiny edge router may *cache* the association between the destination address and the atom ID for a limited period of time to avoid subsequent identical lookups.

Requesting membership information as we just described is one possible way in which a tiny edge route can avoid carrying a full membership table. In addition, there are alternative mechanisms that allow a tiny edge router to receive the membership information it requires without carrying a full membership table. For example, the tiny edge router might create a *tiny membership session* (dashed line in Figure 23) with an edge router through which the edge router keeps the tiny edge router updated with a *subscription* (selection) of atoms, rather than all atoms. Instead of requesting a mapping for an IP address destination, as we described above, the tiny edge router requests the edge router to add the atom corresponding to the IP address to its subscription of atoms. After a certain period of time has expired during which the tiny edge router did not need to forward IP packets destined for an atomised prefix in the atom, the tiny edge router requests the edge router to drop the atom from its subscription.

In all these example mechanisms, the provider AS could delegate the task of servicing tiny edge router requests to a dedicated server, rather than to an edge router.

II. ASSIGNING EDGE ROUTER ADDRESSES

In this appendix we discuss how to assign IP addresses to edge routers and how to route these addresses. We identify two conflicting requirements for edge router IP addresses:

- *Requirement R1* — An edge router's address must be routed in such a way that it can be used as one end of a membership session with another edge router.[46] In order that two edge routers may establish an atom membership session, they must be able to address one another, and IP packets that one edge router sends as part of the session must be able to reach the other edge router. Since the membership session supports the atom membership protocol, and since IP packets addressed to atomised prefix destinations *depend* on the membership protocol for correct forwarding, we would introduce a circular dependency if we allowed an edge router E to use an IP address based on an atomised prefix for its end of a membership session. Therefore we require that E use an address based on a *non-atomised* prefix (Section XI-B) for this purpose. Note that the non-atomised prefix need not be globally routed. Instead, it is sufficient for the prefix to be reachable by edge routers with which E peers.
- *Requirement R2* — An edge router E's IP address must be globally reachable by routers other than the edge routers with which E peers, in order that any router may send an ICMP message to E in response to an IP packet that was encapsulated by E (Section XIII and Figure 15). Recall that such an ICMP message is destined to the IP address that E placed in the source address field of the IP packet that triggered the ICMP message. Since receiving an ICMP message from a router that E does not peer with is not critical for supporting the atom membership protocol, E may use an IP address based on an atomised prefix for this purpose, without creating a circular dependency (as in Requirement R1). Indeed, we wish to *avoid* that an edge router makes a *non-atomised* prefix globally reachable specifically for this purpose, since if every edge router were to globally route an additional non-atomised prefix, the global BGP routing table

[46] The same issues apply to a membership session between an atom originator and an edge router. We omit the discussion of this case.

would increase by one prefix for every AS containing an edge router (potentially over 15,000 prefixes).

We resolve these conflicting requirements by assigning two (or more) IP addresses to an edge router, one of which must be part of a non-atomised prefix, and the other may be part of an atomised or non-atomised prefix. The edge router uses the former address as its endpoint of membership sessions with other edge routers, and it places the latter address in the source address field of encapsulated IP packets. Note that such a distinction between different kinds of IP addresses is analogous to the use of *loopback addresses* to support IBGP sessions, versus the use of physical interface addresses to support EBGP sessions [49].

As we mentioned above, the non-atomised prefix that an edge router uses for peering with another edge router need not (see R1) and preferably should not (see R2) be globally routed. We can prevent global routing of such a prefix as follows. For each AS A containing an edge router with an IP address in prefix p_A, and which needs to peer with an edge router in an adjacent AS B, AS A advertises p_A to AS B as a non-atomised route carrying a *NoExport* community [12]. This community ensures that AS B does not advertise p_A to other ASes, and thus p_A is reachable by AS B, but not globally routed. This solution works in the case that two peering edge routers are in adjacent ASes. In a full deployment scenario, we would expect most membership peerings to be between adjacent ASes. In the case that a membership peering is between two non-adjacent ASes, e.g. under partial deployment (Figure 13), more sophisticated *flexible communities* are currently being proposed that allow more precise propagation of prefix routes [34] [2].

Ultimately, we expect edge routers to assume the role of atom originator and so to originate atoms containing prefixes in their AS. An alternative solution which we did not consider above is to assign to the edge router an IP address based on the atom ID[47] that the edge router originates in its atom originator role. Since the address is not part of an atomised prefix, the edge router may use the address as the endpoint of membership peering sessions with other edge routers (R1). Furthermore, since the address is globally reachable, the edge router may safely place the address in the source address field of an encapsulated packet (R2). Finally, since an atom ID is globally routed in any case, assigning to the edge router an IP address from an atom ID does not require the edge router to add an additional prefix to the global BGP routing table. This solution is easier to manage than the solution we proposed above, but it forces the routing properties of the edge router's address to correspond to the routing properties of one of the atom IDs originated by the edge router, which may be undesirable.

[47] Recall that the atom ID is a prefix; the edge router would receive an IP address from within this prefix.