Technical Report

UCAM-CL-TR-849
ISSN 1476-2986

Number 849

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

SNA: Sourceless Network Architecture

Marcelo Bagnulo Braun, Jon Crowcroft

March 2014

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

SNA: Sourceless Network Architecture

Jon Crowcroft
University of Cambridge
Cambridge, UK
jon.crowcroft@cl.cam.ac.uk

Marcelo Bagnulo Braun
UC3M
Madrid, Spain
marcelo@it.uc3m.es

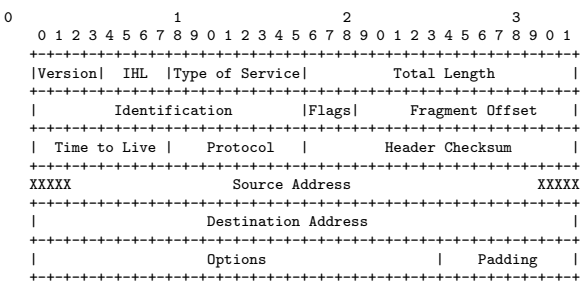
March 17, 2014

Abstract

Why are there source addresses in datagrams? What alternative architecture can one conceive to provide all of the current, and some new functionality, currently dependant on a conflicting set of uses for this field. We illustrate how this can be achieved by re-interpreting the 32-bit field in IPv4 headers to help the Internet solve a range of current and future problems.

1 Introduction

In this paper we discuss SNA, the Sourceless Network Architecture. We ask why there are source addresses in datagrams, and respond with the surprising answer that they are both unnecessary, and a disincentive to the proper confrontation and solution to some of today's problems in the Internet. We then illustrate how one can then move on to start implementing and deploying some future enhancements to the Internet more easily. We acknowledge that we are by no means the first people to propose this idea, but believe that the contribution here is to do so in the context of IPv4[1].



Example Internet Datagram Header

Remember this (RFC 791)? Let's think about the line marked by Xs. So why is there a source address there? The naive answer is that some receivers might want to reply. It's a Datagram, Stupid. Not all higher layers want to send something back! The end-to-end argument says that we do not put a *function* in a lower layer, unless it is required by the majority of upper layer users. We have at least one simple example of an Upper Layer Protocol, UDP, that doesn't *want* to reply. Of course, there are users of UDP that may, for example RPC protocols, and possibly

RTP[1], but that isn't visible in the transport layer, so we don't have a compelling argument to identify the source in the network layer.

More importantly, the source of a packet is not the necessarily the source of a flow. Leaving aside NATs for now, by the time recipient gets a given packet, what makes you think true source is still *where it said it was*, allegedly indicated by that IP source address field? Indeed, NATs mean it isn't/wasn't even *there* there in the first place. The *function* of the source address field, as the indication of where to respond to is neither sufficient, nor even necessary.

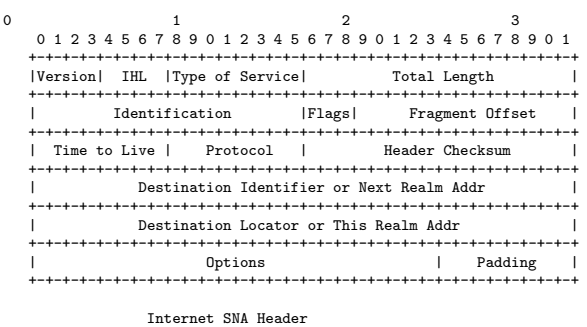
In the meantime, as well as the desire for clarity and simplicity in the network architecture, there are practical pressures on the address space that we need to solve if we are to continue have ever larger numbers of devices always reachable from everywhere, all of the time[2].

This requires more bits.

If we have enough bits more, then there are temptations to divide these bits to get separation of functionality cleanly. Hence when confronted with as many, say as 128 bits, some people want to do $X + Y$, where X is used for one thing (identifier, for example) and Y for another (location, for example).

Indeed, IPv4 addresses conflate *functions* in a number of horrible ways, including at the very least:

- Routing hints
- Interface Specification
- Part of the Transport Multiplex Id
- Flow identification, as in the 5-tuple, used for various service differentiation tricks
- Ingress Police Key - by implication, a source IP address is the identifier for accountability[3].



The idea of $X + Y$ addresses started with O'Dell's proposal[7] for $8 + 8$. What we're about to propose is the moral equivalent of $4 + 4$, much as in the spirit of identifiers used in LISP[6]. However, for compatibility with today's Internet, one can also interpret the fields differently at different places and times. In some sense, the conflation of *function* present in today's address field, is now being distributed over various nodes: another valid interpretation of the new SNA model is that each of the two destination fields is a realm-specific routeable address. Thus the "traditional" destination address can be used to route to egress point in realm 1, then we swap

[1]We note that secure and persistent RPC schemes have session protocols, and that RTP has its own mechanism to identify sources (especially since it was designed to withstand multicast), including authentication. We'll come back to this in the context of TCP later.

[2]There are good arguments against permanent, pervasive, global reachability but we are assuming that these are outweighed by the sheer total numbers of devices with *traditional* service needs, that people seem to by deploying, even if the fraction of dynamically reachable devices is also growing

the value fro the old source field into the destination field, and optionally rewrite the source now to be source of inter-realm router in realm 2 (so that a traditional recipient in realm two can answer). We'll see next that we make use of other mechanisms for SNA-aware hosts running Upper Layer Protocols that feel they need to answer. We are not proposing ROFL[4].

So why are there source addresses in datagrams?

We have 2 32 bit fields that can be routeable IP addresses in different realms or could be ID+Loc (ID could be HIP or other), or actually anything you want..

Note this is novel. It is not like the current other techniques of getting from one realm to another such as translation and tunnelling (a.k.a. *map* and *encap*).

At this point, we'd like to also introduce the terms `key` and `value`, which could get us away from worrying about whether the identity is actually an identity (it depends on the users' security needs) and the location is actually a location (it depends on the context). This will also remind people that there are technologies other than hierarchy and flat tables for mapping between keys and values (and reverse mapping) when there is *churn*.

2 Consequences of SNA

A crucial reason for SNA is that it forces transport protocol writers to confront the problem (either within the transport, or in a shim, or in a proxy) of identifying a multiplex, independent of end-system location, in a secure and timely way. We will talk about this more later.

Benefits of SNA now include:

- Get 2^{32} Internets right now
- Packets still forwarded by all core routers and around 60% of ingress routers
- Can do mobile right immediately
- Can do multi-homing right immediately
- Can do multipath right immediately with an *articulation point* (inter-realm) that is visible to end system (unlike in map or encap)
- Don't need IPv6 ever!

Consequences that must be dealt with immediately:

- Transports that do want to send/get an answer, such as TCP, UDP based RPC-like transports (DNS, SNMP and NFS), SCTP, DCCP, RTP/UDP.
- ICMP, Mistakes, errors, bad stuff
- Ingress Policing
- Net to End signalling in general

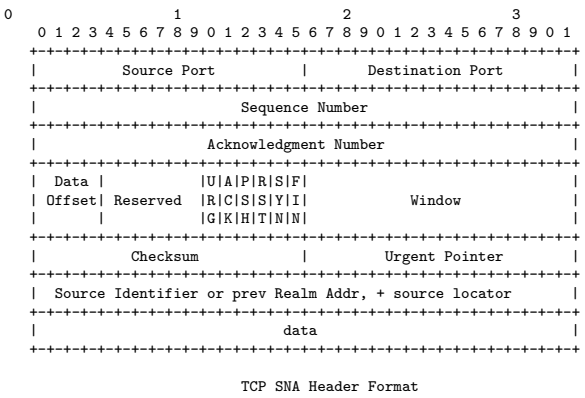
2.1 Transports that do want to answer

We argue transports that wish to respond, need to answer a transport entity. To do this, they need to respond to a corresponding network identity, in the first instance, and not a location. Of course, we wish to make the response efficient. However, we also need to cope with the possibility of source and destination moving during the connection establishment phase.

Furthermore, we can implement lite with Shim, proxy, or new transport no-op. Much of the shim6,six-one[11][12], and HIP work applies immediately

In a resource rich world, the Transport Setup could send a Handle used to lookup key-value $4+4$/SNA source from client to server. One could send an FQDN in the SYN packet, then the receiver would use this to look up the source location/identifier $4+4$ pair. However, this level of indirection is excessive overhead and doesn't solve the race problem above at all (indeed just introduces a dynamic DNS update overhead and synchronisation problem as well).

Instead, we put the SNA source in the transport options. both in the TCP SYN option, and in subsequent data packets.



TCP SNA Header Format

Of course, we still need to put this information in to the DNS to allow SNA-capable hosts to lookup $4+4$ addresses, and we still need to update the DNS to allow new hosts to reach servers that move. This is very much in the spirit of the IPv6 mobility work. Indeed, we can have a number if agents to help.

Dyn-DNS, LISP or a Mobility Agent, or other new Server can return FQDN-SNA locator Mappings. We discuss the nature of these servers later. This is not a new idea, having been discussed in IPNL[8] and to which we owe a very large part of the architecture here. Improved naming and addressing architectures themselves have appeared with increasing frequency for example, FARA[5] and NIRA[13]. However, more simply we can build a NAT-like service, which we call a SNARK (SNA Re-Key). This functions as NAT+Swap and serves as a limited form of indirection, used for interworking with legacy nodes, and for temporary routing while one or two end points are in the process of moving and have not yet delivered a packet to the other end to inform it of the new location which can be routed to directly, as described variously, for example[10][9][2].

Subsequent packets can also use cookies, as with SCTP (not SYN Cookies) to indicate the speed up the multiplex to the receiver;

Packets are routed in this realm (or end-to-end when both ends are SNA hosts) on the destination locator field in v4 forwarding devices, and on the 64 bit field in SNA networks.

Four Interworking cases for end systems (transport) need consideration for deployment:

1. IPv4 to IPv4

   This is a no-op. The current system works.

2. SNA to IPv4

   The host receiving an SNA TCP setup will not respond with the option enabled, so SNA reverts to IPv4 and case 1.

However, if the receiving host is in another realm, SNA will need to route via a relay. If the relay operates as below where we describe the SNA-RK[3], then the receiving host will setup an IPv4 TCP standard connection with an apparent destination at the relay, while the SNA host will see the connection established as far as the relay, which will then use cookies to detect SNA packets, and decide to do the re-key operation (swap source into destination, and re-write now IPv4 source field with own next-realm address) and forward re-keyed IP packets efficiently.

This SNA clients can reach IPv4 legacy servers in the same of a different realm, via an SNA-RK.

3. IPv4 to SNA

   This works transparently, which is an attractive feature of SNA.

4. SNA to SNA.

   Four cases (recurring the four cases of which this is fourth). First, where the net is IPv4. Routing happens on the destination field, so no network change, but the listening host (SNA capable TCP) needs to look in the SYN OPT, and lookup the senders SNA location and identifier. It then can send back the SYN ACK, and after the three-way handshake, packets carry to be sent as normal.

   Second and third case, the SNA hosts are in different IPv4 realms. Requires an SNA-RK. and participation of the mapping service.

   Fourth case is the SNA Internet, where the routers are routing on destination only, and all the 32 bits of destination are for routing, freeing the other 32 bits for identifiers, and the network can scale quite a lot further since we no longer need the low order bits for host as we did in IPv4. This does not preclude the continued use of NAT technology (now subsumed within SNA-RKs) for other (e.g. access control/privacy) reasons as well as legacy support.

The next three cases are concerned with support of novel network services which may offer more efficiency if we have SNA aware transport:

- Mobility

  To avoid triangular routing, we need end points to inform each other of a move. Since an SNA transport multiplex doesn't depend on the locator, this can happen *mid-connection* painlessly, so long as there is a SNA-RK available to route packets temporarily via while the locator for one or both ends is changed. The various handover techniques for doing this work. The end points need to interwork with the locator allocation, and the locator-identifier service needs a mapping updated so servers that move can be reached by new clients. We discuss the nature of the update problem later.

- Multi-homing

  A multi-homed SNA-aware host simply has multiple locators but one identifier. SNA-aware transports might use this if they wish. A mapping service (could interface via LISP) can advertise this). Again if a host wishes to modify the locations available, the mapping service needs to allow appropriate update, although mid-flow updates are easy in SNA.

[3]Astute readers will observe that this is going to be a type of NAT, but also that we have to be able to hunt for it somehow, hence hunting the snark.

- Multipath-Aware SNA

  Multipath-aware transport stripes data over multiple paths. SNA makes this fairly easy. One can use multiple SNA-RKs. ISPs may wish to control how these places in the net are advertised so that the multipath routing system affords users multiple paths that align with intra-domain and interdomain traffic engineering, and inter-domain policies.

2.2 What are the problems with ICMP

We note that ICMP is layered *on top of* IP. Architecturally then, we make two decisions: firstly, ICMP can reasonably be expected to be SNA aware; secondly, a better place to send advisory information in the future Internet is *onwards* to the receiver, not *backwards* to the sender. Obviously the unreachable cases need to be considered more carefully, although if an end system is not reachable, a SNA-RK might be still reachable.

ICMP implementations (in routers) are not only not on the fast path, they are almost always rate limited, and frequently filtered. Routers also usually implement TCP or other more complex end-to-end protocols (e.g. for BGP sessions or for Web based management) so the machinery of SNA can be reasonably expected to not exceed router software capability.

Nevertheless, there are four important(ish) cases to consider:

1. Redirect - is link local so trivial.

2. Echo - is an *application* so we propose re-implementing it as SNA-aware (in any case, it is not on fast path)

3. Errors (unreachables) - were always a bad idea (except maybe port unreachable, which is an application/transport

4. MTU discovery - can do by sending fragsizeexceed to the *destination*. Indeed this would also ease MTU discovery for multicast applications, and will allow transport to make sensible choices in heterogeneous multipath cases.

In legacy case (if you must) a source can always start with a few IPv4 source address (locator) packets intermingled with the new SNA packets, just to elicit some of the required answers (e.g. MTU discovery, or traceroute legacy support).

Note, misdirected ICMP errors can be handled, because they include sufficient of the original packet that caused the problem, that an unintended receiver can (and will in most popular known end system OSs, that we have examined) discard

2.3 Ingress Policing/Spoof Detection

We note that a fairly high fraction of the ASs in the Internet now actually do implement RPF checks on source addresses, viz@ `http://spoofer.csail.mit.edu/summary.php`

However, we claim that ingress policing on identity keys is far more useful since it associates misbehaviour with an auditable or accountable entity, as discussed elsewhere[3]. Indeed, bad guys can easily steal an address from someone else on a LAN and jam traffic from the *authentic* host, while sending evil packets, so an identity based ingress policer is a better approximation to the security solution to the threat.

In many cases, we can do this easily at ingress link local (or there is only one host on xDSL line :)

So the point here is that when you have transport information, you

have a source identifier that matters i.e. anyone you want to DOS attack will ignore you if you don't have an identifier, in the new packet; so only if you put one there do you get to go towards source, but you need a source id to be TCP-SNA compliant so you give away your id: so game over.

3 Multi-* transport - things this makes easier

So I now have a transport multiplex with 2 64 bit id+loc but can wildcard the loc part to do multi-* (for now, can put it 1 hop away too i.e. in access router, and proxy for the state - similar to TCP header compression):)

- SNA permits seamlessly mobility, at least in the same way as Mobile IPv6 with route optimization.
- SNA allows multi-homed transport, and have the sub-flows to/from a given interface be indicated by varying the routing part of the SNA.
- For the same reason this allows explicit (signalled) multi-homed TCP or other transport, it also allows a single end-to-end flow to indicate which sub-flows are mapped to which sub-paths on a multi-path route. This may be useful to ISPs, since recent theory on congestion control and multipath indicates that it is beneficial to traffic engineering. it could be very useful for TFRC compliant multimedia flows since failure of sub-paths only degrades the overall session, rather than disrupting it during route re-convergence.
- Additionally, each end gets to see which part of the multi-path each packet arrived over (or is lost or ECN marked)

3.1 Doesn't this break Multicast IP?

It seems that SNA (if used) seems somewhat to break multicast. Multicast currently uses source address as hook to build trees. Thus multicast routing would have to be SNA aware, and map an identified to a locator and build a locator-based tree. Alternatively, Multicast could just stay with IPv4 source address, which is fine because most multicast applications are RTP based, and RTP isn't broken the way TCP is w.r.t. what it thinks is the originator of data.

However, RTP/UDP multicast (and unicast) applications work correctly if source changes or traffic is multipath, so they don't need the SNA change to incentivize people to fix them[4]!

4 The Identity to Location Key-Value Mapping Problem

As we hinted above by using the magic term {Key,Value}, the hardest part of all is that this needs a service that provides a mapping from Identity to Location. In the current world, the DNS maps FQDNs to IPv4 addresses.

[4]We also claim that we are not mandating SNA - it is merely a different use of the same headers. Applications can still use IPv4. Examples where this may be most useful are where the traffic is largely nearly all local such as end host DNS or RPC, or RIP exchanges over UDP between neighbour routes.

However, this doesn't actually solve any of the problems that arise when you want to *use* an architecture that explicitly separates out identity and location for any of the purposes outlined above (auditable sources, mobile source and destination, multi-homed applications, and multipath transport).

The reason this is a problem is that the service must be low latency, and must reflect changes in the mapping rapidly enough for seamless mobility, for both forward (key to value) and reverse (value to key) lookups. This means that neither of the current approaches using a database of the form of either a router-centred service, or a DNS centred service, with either pull or push update, will work. The problem requires one to refactorize the database in the face of update - in other words this is the first occurrence of a *data driven* networking problem that is not an application for end users, but is a part of the network itself.

Some of the solutions to large scale data-driven networking that are being employed in online social networks such as recommendation networks might work very well here to manage the typical churn rates in such a service that one would expect now that there are $3 * 10^9$ mobile nodes compared with only $1 * 10^9$ fixed Internet hosts.

5 Security Considerations Harmless, Mostly

We believe that the security considerations are somewhat less stringent than for HIP, since the timescales for trust that the SNA is not undermined by man-in-the-middle exploits on routers are those of a single transport session only.

However, it certainly is an important part of future work to carry out a proper threat analysis. In a purist sense in the end-to-end race, Source addresses shouldn't be in the net layer because not all ULPs want them Giving away your source IPv4 to any tom, dick or harry (or Alice, bob and carol) seems a bit careless. Indeed, others have already argued source addresses can be considered harmful for security, for example in `http://www.tml.tkk.fi/~pnr/publications/nordsec2001.ps`.

One criticism of SNA is that the lack of symmetry in location of from/to fields means potentially that debugging the net just got slightly harder. On the other hand, the impact of mobility (and multi-homing and multipath) on network complexity already required enhanced tools for tracing where things are going wrong. We think that on balance, SNA could actually make that easier.

Finally, it is clear that SNA as it stands breaks IPSec.

6 Evaluation

There is no obvious performance impact on (uni-path) data forwarding from SNA, but there is increased control plane (ICMP) implementation complexity. There is also increased complexity in end hosts in de-multiplexing packets and in the size of the TCPCB.

From an application programmers perspective, to make SNA transparent, i.e. so that SNA-TCP has the same socket API, we require it to internally look up a destination SNA/IP in the DNS asking for an SNA "address" (this is mapping an SNA id onto an SNA location). This needs the DNS to store a new RR which is basically an SNA address. When a DNS server responds to resolver queries for the SNA for an FQDN, it returns the SNA loc + the SNA id; legacy lookups for SNA host names just get an IPv4 address, which just looks like the SNA id in most cases for SNA hosts.

There is a new TCP SYN SNA option which puts the source's SNA in the data of the TCP SYN option. An SNA-TCP in listen state, receiving a TCP with the SNA option, stores the

source field from the incoming SYN. so the SNA TCP PCB now has to store 4 32 bit fields at each end the SNA of each end is 2 32 bit fields...a locator and id. This is no different than transport tunnels for mobility.

Mobile/multihome/multipath TCPs need to interact with IP to get upcalls about incoming SNA-IP packets that changed or we moved or an interface went up/down, so we can add/delete entries in the DNS for the SNA locators. There are colleagues working on projects to define TCP (and other transport) behaviours and mappings for multipath and multi-homing.

Finally there's a thing that looks like a NAT (the aforementioned SNARK) which provides interworking between SNA client and IPv4 server - (and some inter-realm cases). so this does one of two things depending on what color snark it is[5]: swap src/dst fields, and replace src with snark's address on outging interface towards new destination; if providing mobility support, a SNA-RK so it swaps src/dst, but doesn't overwrite src providing rendezvous, which doesn't obviate later route optimisation.

7 The Future

We are working on an implemention of SNA for host and router. Specifically, the BSD code entails changes in the following places. Many of these are minimal.

```
ip_input: reject loopback address and network source addresses
ip_input: implicitly used in checksum calculation
ip_reass: use ip_src in fragment reassembly
ip_forward: use ip_src to target routing redirect
udp_input: safe in sockaddr_in for user application to use
udp_input: used in checksum calculation
udp_input: checked for multicast socket delivery if socket isn't using
   INADDR_ANY
udp_input: checked in unicast socket delivery if socket isn't using INADDR_ANY
udp_ctlinput: matching ICMP to a socket to deliver an error
tcp_input: used in checksum calculation
tcp_input: connection lookup for a fully matching socket, listen socket, or in
   the IP forward case where we're intercepting and terminating a transiting
   TCP session for a non-local address
tcp_input: initializing new connection state
tcp_input: in specifically rejecting a connection to self for a new connection
tcp_input: in checking for multicast and broadcast sources for a new
   connection
tcp_dropwithreset: in avoiding sending a RST to a broadcast or multicast
   address
```

We need to evaluate fully the cost of the ICMP router changes, and carry out a proper security analysis.

8 Acknowledgements

Thanks much to Robert Watson for the code. Thanks for comments to the Trilogy Project and IMDEA researchers, Richard Black, at MSR on complexity, Mark Handley at UCL's unpublished 4 + 4 draft, Bob Briscoe for anti-spoofing BCP-compliance, Tim Griffin on ICMP; KK on multicast, and RFC2110.

References

[1] ad Pekka Nikander, C. C. Ipv6 source addresses considered harmful. In *Proceedings Sixth Nordic Workshop on Secure IT Systems* (2001).

[5] don't get me started on boojums:)

[2] Ahlgren, B., Eggert, L., Ohlman, B., Rajahalme, J., and Schieder, A. Names, addresses and identities in ambient networks. In *DIN '05: Proceedings of the 1st ACM workshop on Dynamic interconnection of networks* (New York, NY, USA, 2005), ACM, pp. 33–37.

[3] Bender, A., Spring, N., Levin, D., and Bhattacharjee, B. Accountability as a service. In *SRUTI'07: Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet* (Berkeley, CA, USA, 2007), USENIX Association, pp. 1–6.

[4] Caesar, M., Condie, T., Kannan, J., Lakshminarayanan, K., and Stoica, I. Rofl: routing on flat labels. *SIGCOMM Comput. Commun. Rev. 36*, 4 (2006), 363–374.

[5] Clark, D., Braden, R., Falk, A., and Pingali, V. Fara: reorganizing the addressing architecture. *SIGCOMM Comput. Commun. Rev. 33*, 4 (2003), 313–321.

[6] Meyer, D. Lisp, March 2008.

[7] O'Dell, M. Gse-an alternate addressing architecture for ipv6, 1997.

[8] Ramakrishna, P. F. Ipnl: A nat-extended internet architecture. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2001), ACM, pp. 69–80.

[9] Snoeren, A. C., and Balakrishnan, H. An end-to-end approach to host mobility. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking* (New York, NY, USA, 2000), ACM, pp. 155–166.

[10] Stoica, I., Adkins, D., Zhuang, S., Shenker, S., and Surana, S. Internet indirection infrastructure. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2002), ACM, pp. 73–86.

[11] Vogt, C. Six/One: A Solution for Routing and Addressing in IPv6.

[12] Vogt, C. Six/One Router: A Scalable and Backwards Compatible Solution for Provider-Independent Addressing. *ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch)* (Aug. 2008).

[13] Yang, X. Nira: a new internet routing architecture. In *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture* (New York, NY, USA, 2003), ACM, pp. 301–312.