Cutting Down on IP Address Waste

IPv4 Unicast Extensions Project

IPv4 Unicast Extensions Project

- An effort to reduce waste of IPv4 addresses that are currently completely unused
- Established by John Gilmore, with technical work by Paul Wouters, Dave Täht, Seth Schoen
  - Mike Karels has also joined as co-author of one draft RFC
- Thanks to many colleagues who have offered comments and historical insights

Our proposals

- Unreserve four kinds of reserved IPv4 address, asking implementers to treat them as unicast
  - These addresses are reserved for historical reasons, to minimal or no useful purpose today
  - This will free up a substantial amount of IPv4 space
- With the measurement community, test the effects of using these addresses on the Internet
  - If useful, they can be allocated some day

Historical decisions

- Throughout the 1980s—when IP’s future was less clear, and scarcity a less prominent concern—various decisions treated large numbers of addresses specially
- With decades of hindsight, some of those decisions are not helpful and are now preventing large amounts of otherwise useful address space from being used for unicast addressing

Current status

- Four Internet-Drafts proposing to unreserve addresses for unicast use
  - draft-schoen-intarea-unicast-lowest-address
  - draft-schoen-intarea-unicast-240
  - draft-schoen-intarea-unicast-0
  - draft-schoen-intarea-unicast-127
- Presented first two at IETF112 to some controversy (especially about 127/8!)
- We’ll discuss specific addresses and software support

Wasted addresses: Lowest

- Suppose we have a network 42.43.44.0/24
  - Berkeley chose the *lowest address* (42.43.44.0) for broadcast
  - Developers elsewhere chose the *highest address* (42.43.44.255) for broadcast
- The highest address won out in all recommendations and documentation, but the lowest address remained reserved, explicitly for backwards compatibility

  … with systems that haven’t existed for decades!

Wasted addresses: Lowest

- One address is wasted per subnet. For example, if you have a /28 subnet, the duplicate broadcast address would be 1/16 of your addresses
- Subnetting within an organization causes this to become more significant, as each subnet loses an additional address

Lowest address fix is local (!!)

- **Under existing RFCs**, distant (non-subnet-local) hosts must not assume the netmask of your hosts (they don’t know where subnet boundaries fall in networks to which they’re not attached)
- If just **your router and LAN** support the lowest address as unicast, the rest of the Internet should already interoperate with the lowest address on your subnet!
  - Try examples at http://ec2-reachability.amazonaws.com/

Wasted addresses: Experimental

- All the addresses from 240.0.0.0 upward (2^{28} addresses) are “reserved for future use” due to a decision in 1983
  - Futureproofing IPv4 for potential new addressing modes (e.g. dedicated anycast or encoding >32-bit addresses)
  - That was reasonable at the time, but 240/4 has *still* never been used for anything
  - New IPv4 addressing modes are very unlikely to be invented now

Wasted addresses: Zero network

- All the addresses from 0.0.0.0 to 0.255.255.255 (2^{24} addresses) are reserved due to a decision in 1981
  - Mainly intended to be used for autoconfiguration
  - But the autoconfiguration solutions that won out (BOOTP → DHCP) use only *one* of these addresses (0.0.0.0), not 2^{24}; the system that would have used all of them was deprecated in 1989

Wasted addresses: Loopback

- All of the addresses from 127.0.0.0 to 127.255.255.255 (2^{24} addresses) are reserved due to a decision in 1986.
  - All of these mean “this system”
  - By contrast, IPv6 only has the single loopback address ::1
  - It’s not common for loopback addresses outside of 127.0.0.0/16 (65536 addresses) to be used at all
    - Apparently one VPN product in Japan uses them

Wasted addresses: Multicast

- All of the addresses from 224.0.0.0 to 239.255.255.255 (2^{28} addresses) are reserved for multicast, which is little-used compared to unicast
  - In principle, the majority of these addresses can be reclaimed: they'll never be used for multicast services
  - Reclaiming these addresses is more complicated because special behavior is still ubiquitous, and used/unused blocks are more interleaved compared to other ranges
    - We don't currently have an I-D about this address space

How many addresses?

draft-schoen-intarea-unicast-lowest-address

- “One address per subnet, Internetwide”

draft-schoen-intarea-unicast-240

- 2^{28}-1 = 268,435,455 (6.25% of all IPv4)

draft-schoen-intarea-unicast-0

- 2^{24}-1 = 16,777,215 (0.389% of all IPv4)

draft-schoen-intarea-unicast-127

- 2^{24}-2^{16} = 16,711,680 (0.389% of all IPv4)

hypothetical 224/4: hundreds of millions

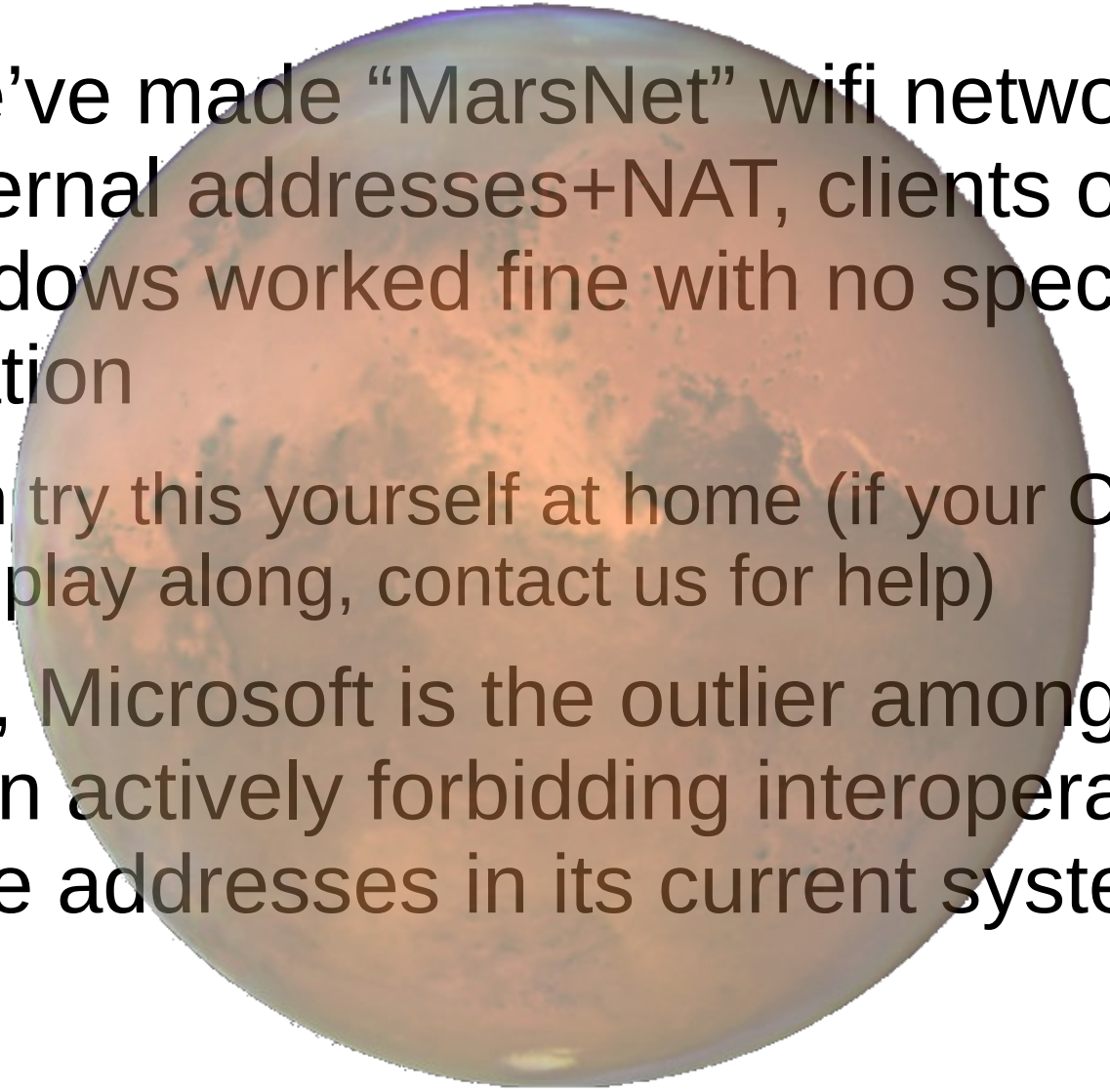
Software support

- **240/4** : Most popular Unix-based systems (mostly inspired by a prior proposal in 2008!), including Linux, Android, macOS, iOS
- **Lowest address** : Linux, FreeBSD
- **0/8** : Linux
- **127/8** : None known
- **224/4** : None known
  - Changes mostly consist of identifying and *removing* special cases in IP stacks, and testing interoperability
  - Generally, no one has noticed

No one noticed?

- Right!
- Many of the changes we propose landed in various operating systems already (through our and others' work)
  - There was no catastrophe
  - We have yet to find any complaints or bug reports
- You may be watching this presentation on a 240/4-capable device right now!

240/4 experiences



- When we've made "MarsNet" wifi networks with 240/4 internal addresses+NAT, clients other than Windows worked fine with no special configuration
  - You can try this yourself at home (if your CPE router doesn't play along, contact us for help)
- Currently, Microsoft is the outlier among OS vendors in actively forbidding interoperability with these addresses in its current systems

A gradual process

- Problem: if machines A and B disagree about the validity of an address, and one is numbered with that address or asked to route it, communication may not occur
- It takes time to update software
- Our changes have limited backwards compatibility (except for lowest-address), so getting widespread support in devices will take some time
- That’s why we should start in 2008; if not then, now

When to plant a tree

"The best time to plant a tree was 30 years ago. The second best time is now." – Proverb

Measurement

- We’d like to work with the Internet measurement community to get some large-scale metrics about usability of reserved addresses
- Both now and following, or as part of, Internet community consensus on trying to make reserved address space more useful
- Empirical data can inform the later decision to allocate historically reserved address space

Debogonization

- Cloudflare got official permission to use 1.1.1.1 for a DNS server, launched in 2018
- Many networks had hard-coded blocking this range. Cloudflare took > 1 year investigating users' reports of unreachability and working with ISPs to remove blocks
  - But following that, 1.1.1.1 is now extremely widely reachable on the Internet (still not 100%, but very high)
- We believe we can follow a similar process with formerly reserved addresses, once software support for them is widespread by default

Concerns

- We’ve heard a number of concerns from the community, at IETF and on NOG mailing lists
- Most were about our 127 draft (which got a lot of publicity!), but some were broader
  - Note that technical concerns about different reserved addresses are *different* — for example, you can unilaterally use the lowest address by patching only your own systems
- Let’s talk about a few common objections

Relation to IPv6?

- Most common objection: “why not just use IPv6”?
  - We favor IPv6 adoption, but still want to mitigate IPv4 address scarcity
    - IPv6 as a complete replacement for IPv4 is still decades away (if not “never”)!
    - Meanwhile, IPv4 address scarcity pain continues to exist
  - IPv6 software support is already widespread and mostly excellent; *ISP* support is lacking
    - This makes “IPv4 vs. IPv6” a false dilemma, especially from the point of view of people working on software

IPv4 maintenance?

- Some people say nobody should maintain or improve IPv4
  - We find this idea seriously mistaken
    - IPv4 is the lingua franca of the Internet, the most-used network layer protocol in the world
    - It still carries most Internet traffic
  - There’s no IETF consensus to stop work on IPv4; none emerged from sunset4 WG (2012-8), which proposed deprecating v4 or forbidding maintenance
    - IETF policy requires v6 compatibility, not neglect of IPv4

Immediate exhaustion?

- Some people worry that a new RIR allocation of (e.g.) 240/4 would be consumed just as quickly as the last /4s allocated by RIRs
  - Market prices, now around $50/address, are strongly incentivizing more efficient uses of newly available address space
  - APNIC was willing to use market mechanisms itself in allocating limited address space
  - Demand is huge, but quantity demanded at market prices or at $0 is substantially different

Inconsistent behavior?

- An objection to Fuller, Lear, and Meyer's 2008 proposal in 2008, repeated against our proposals, is that we can no longer change addressing semantics because hosts will then disagree about whether an address is valid or what it means
- But implementations **already** disagree about whether addresses are valid; that's the status quo!
  - If we want to reconverge host behavior, standardizing the very widespread acceptance of 240/4 is the most efficient path to do so

Testing

- We’ve been testing the behavior of individual operating systems and routers with regard to reserved addresses
- We’d like to start testing the use of these addresses on the Internet together with the Internet measurement community
- We anticipate that it will be years before these addresses can readily be allocated like other unicast addresses—and that they will probably still be useful at that time

Economic value

- IPv4 addresses have enormous economic value; hundreds of millions of dollars at a time have been paid for large blocks
- Some people express the hope that the cost of IPv4 addresses will continue to grow and that this will encourage IPv6 adoption
  - We think this connection is typically more tenuous than people hope

Thanks!

- Questions or comments?
- Contact us:
  - Seth Schoen <schoen@loyalty.org>
  - John Gilmore <gnu@rfc.toad.com>

BONUS SLIDES

On topics or questions that may arise

Thoughts on updating the Internet

- There are a lot of problems on the Internet involving unupdated software. E.g., DST X3 Root certificate expiration in October 2021 → Let's Encrypt certificates now mostly only work for clients updated since 2015
  - This is an Internet-wide problem associated with every non-backwards-compatible change (cf. "Caniuse")
  - Arguably, it's getting less severe these days, as automatic software updates become *much* more common
  - But people still use devices that are in EOL (although this is typically a big security problem and is one factor in the prevalence of botnets) (and many cheap phones get few or no OEM software updates)

Needed patches are very small

- Typically, remove code or macros that create special cases for an address range; the IP implementation will then default to treating it as unicast.
- That is, we typically just have to *stop* treating these address ranges as an error, and then they will work!
  - Sometimes also a small userspace patch, so tools like `ifconfig(8)` won't complain about reserved addresses

Linux lowest address

```
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -1122,10 +1122,8 @@ void fib_add_ifaddr(struct in_ifaddr *ifa)
                                        prefix, ifa->ifa_prefixlen, prim,
                                        ifa->ifa_rt_priority);

-               /* Add network specific broadcasts, when it takes a sense */
+               /* Add the network broadcast address, when it makes sense */
                if (ifa->ifa_prefixlen < 31) {
-                       fib_magic(RTM_NEWROUTE, RTN_BROADCAST, prefix, 32,
-                                       prim, 0);
                        fib_magic(RTM_NEWROUTE, RTN_BROADCAST, prefix | ~mask,
                                        32, prim, 0);
                }
```

Linux zero network

```
--- a/include/linux/in.h
+++ b/include/linux/in.h
 static inline bool ipv4_is_local_multicast(__be32 addr)
@@ -67,7 +72,7 @@ static inline bool
ipv4_is_all_snoopers(__be32 addr)


 static inline bool ipv4_is_zeronet(__be32 addr)
 {
-       return (addr & htonl(0xff000000)) == htonl(0x00000000);
+       return (addr == 0);
 }
```

Linux 240 (“Class E”)

```
--- a/include/linux/in.h
+++ b/include/linux/in.h
@@ -262,9 +262,10 @@ static inline bool ipv4_is_local_multicast(__be32 addr)
        return (addr & htonl(0xffffff00)) == htonl(0xe0000000);
 }

-static inline bool ipv4_is_badclass(__be32 addr)
+static inline bool ipv4_is_lbcast(__be32 addr)
 {
-       return (addr & htonl(0xf0000000)) == htonl(0xf0000000);
+       /* limited broadcast */
+       return addr == INADDR_BROADCAST;
 }
```

→ Plus s/ipv4_is_badclass/ipv4_is_lbcast/g in other kernel source

Linux 127 (“loopback”)

```
--- a/include/linux/in.h
+++ b/include/linux/in.h
@@ -37,7 +37,7 @@ static inline int proto_ports_offset(int proto)

 static inline bool ipv4_is_loopback(__be32 addr)
 {
-       return (addr & htonl(0xff000000)) == htonl(0x7f000000);
+       return (addr & htonl(0xffff0000)) == htonl(0x7f000000);
 }
```

→ Plus userspace patches to headers, systemd, and ifconfig

FreeBSD lowest address

```
--- a/sys/netinet/in.c
+++ b/sys/netinet/in.c

        return ((in.s_addr == ia->ia_broadaddr.sin_addr.s_addr ||
             /*
-             * Check for old-style (host 0) broadcast, but
+             * Optionally check for old-style (host 0) broadcast, but
              * taking into account that RFC 3021 obsoletes it.
              */
-             (ia->ia_subnetmask != IN_RFC3021_MASK &&
+             (V_broadcast_lowest && ia->ia_subnetmask != IN_RFC3021_MASK &&
            ntohl(in.s_addr) == ia->ia_subnet)) &&
             /*
```

→ Plus boilerplate that makes V_broadcast_lowest a sysctl option

Should IETF maintain IPv4?

- IETF policy statements require IPv6 compatibility everywhere, but *don't* forbid continued stewardship of IPv4
  - From 2012-2018, the sunset4 WG discussed deprecating IPv4 or forbidding IETF to work on maintaining it
  - No IETF consensus emerged to do these things
- Without IETF's stewardship, IPv4 implementations may lose compatibility
  - Or another SDO with less institutional expertise may take over IPv4 maintenance

Continuing to maintain IPv4

- We hope for the reverse consensus: that IETF continues to accept responsibility for maintaining IPv4, the most widely-used network-layer protocol in the world
- We agree that IETF and the Internet community should encourage IPv6 support in all new deployments
  - But, as some sunset4 participants noted, it’s not clear that IETF has much power to affect IPv6 deployment one way or the other!

Carrots and sticks

- We’re surprised that some community members want to pursue making IPv4 worse or harder rather than making IPv6 better or easier
- E.g. people have told us (and stated in sunset4) they feel it would be *bad* to have more IPv4 addresses available … because then people would use them
  - Some also disapprove of IPv4 address markets and efforts to reclaim or improve utilization of legacy blocks

Deprecating IPv4?

- IETF had a long-lived WG called sunset4 which worked on ideas to deprecate IPv4, declare it historic, or ban IETF from doing work to maintain it
  - Stronger than existing requirements to guarantee IPv6 compatibility in new protocols/documentation
- IETF-wide consensus failed partly due to uncertainty on exactly what this would mean in practice