IPv4 Unicast Extensions

draft-schoen-intarea-unicast-lowest-address
draft-schoen-intarea-unicast-240
draft-schoen-intarea-unicast-0
draft-schoen-intarea-unicast-127

Seth Schoen, IPv4 Unicast Extensions Project
November 9, 2021

Our goals

- Standardizing and implementing a few one-line patches can create billions of dollars of value in the existing Internet
- By recycling IPv4 address space **that nobody is using today** with small, gradual changes
- To eventually provide **hundreds of millions** of new unicast addresses

Running code

- The largest change proposed here has been **default behavior** in widely-used operating systems for 13 years
  - With no known complaints or problems
- Other changes proposed by our drafts are the default behaviors in Linux and FreeBSD systems

Reserved address overview

- Many addresses were reserved between 1981 and 1986, when address scarcity was not a problem
  - At that time the Internet was very much an experiment. Its inventors could not know what a success it would become
- 30 more years of experience shows that many of **these reservations are no longer needed**
- These reserved addresses remain unused in a time of acute unicast address shortage. They have enormous economic value and can be useable with extremely small software changes.

draft-...-unicast-lowest-address

- The lowest address in each subnet (e.g. 93.184.216.0 in 93.184.216.0/24) is a duplicate broadcast address
  - 1983's 4.2BSD implemented broadcast there, before it was standardized at the highest address of each subnet.
  - Everyone agrees this is deprecated ("non-standard" per RFC 1122; "obsolete" per RFCs 1812 and 3021). 1986's 4.3BSD adopted the standard highest address.
  - Our draft reclassifies it as an ordinary unicast address.

Lowest address implementations

- Unicast on the lowest address is now **default behavior** in Linux and FreeBSD
  - This saves one assignable address per IPv4 subnet, Internetwide
- Under RFCs 1812 and 4632, distant hosts **require no changes** to interoperate with these addresses, since they can’t assume where the subnet boundary falls
- **You get the benefit of this extra address just by changing your own subnet**

draft-…-unicast-240

- The former “Class E” or “Experimental” range (240/4) contains **268,435,456** addresses, over 6% of all IPv4 address space (**i.e. 240.0.0.0 up to 255.255.255.254**)
- These addresses were “reserved for future use” in the 1980s, possibly for future addressing modes like anycast (RFC 1546)
  - This was prudent then, but no such uses have emerged or will emerge.
- Proposals to convert this range to unicast in 2008 did not advance at IETF, but found favor with implementers, and have been **widely supported** by operating systems since then.

draft-…-unicast-240

- Our draft designates this range as unicast, asking every OS to implement this in their software.
- Our draft **does not allocate** this address space.
  - We aren’t deciding, and implementations don’t care, whether it’s ultimately used as public or private address space.
  - Allocation is a separate question and would only happen, if at all, after wide compatibility exists in the field, and after the effects of announcing routes in this range have been tested.

240/4 implementations

- Since 2008, most operating systems have supported 240/4 as ordinary unicast or can easily be configured to do so. This support has created no operational problems.
  - Including Linux, Android, macOS, iOS, many more
- It **works so well** that some cloud vendors are using it unofficially as private address space.

draft-...-unicast-0

- We propose unicast use of 0/8. It was reserved in 1981 for autoconfiguration in RFC 792. In 1989 this was obsoleted by RFC 1122 and has remained **reserved for nothing** up to this day.
  - Today, autoconfiguration is via DHCP, using only 0.0.0.0
- The draft documents current behavior in Linux.
- For future discussion

draft-...-unicast-127

- Convert most of 127/8 from local loopback to unicast, retaining loopback in 127.0/16
  - IPv4 doesn’t need **16,777,216 loopback addresses;** IPv6, for example, has only one (::1)
  - Very little software ever uses or refers to the others
- No released implementations yet
- For future discussion

Required changes are small

- We prepared and tested changes for several environments to implement all these drafts
- Typically, the needed change is **about one line of code** per draft and simply removes a spec ial case in an OS kernel
- Many of these changes are already released in popular operating systems
- Patched systems interoperate well

A gradual process

- Standardizing these proposals would **encourage uniformity** among existing implementations, reducing potential issues
- Making these minor changes now prepares the Internet for a time when standards bodies can allocate these addresses for operational use
  - **“The best time to plant a tree is 30 years ago; the second-best time is now”**
- If adopted, these changes would roll out in **ordinary software updates** over the next several years.

A gradual process (cont'd)

- Our drafts **do not allocate the new unicast address ranges**
  - Specific allocations would take future action by IETF, IANA, or ICANN
- Achieving reachability for formerly impaired addresses is a known problem with known solutions

Conclusion

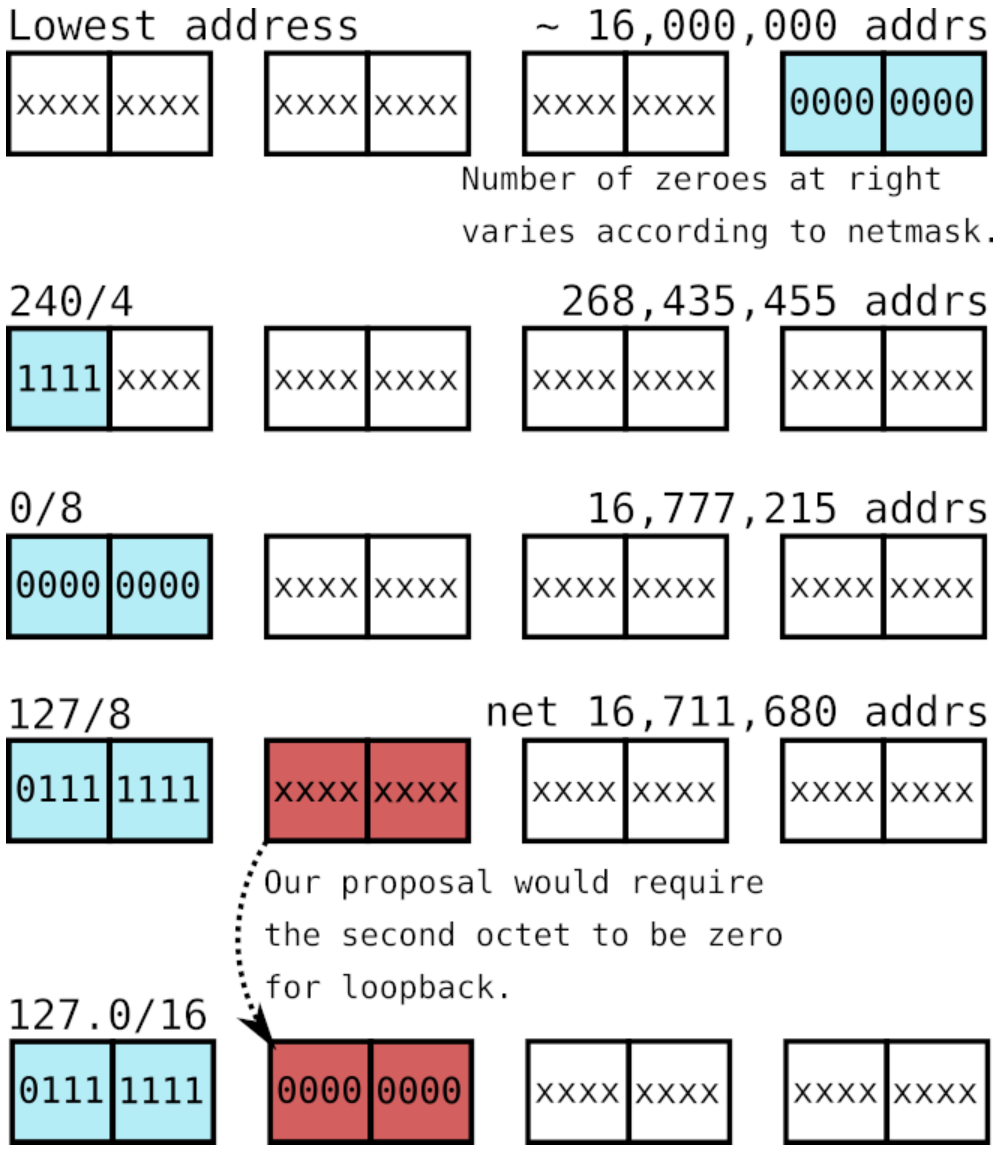
- Standardizing and implementing **a few one-line patches**, at low cost, can create billions of dollars of value for Internet users at large
- Existing implementations prove that deployment is an **incredibly low risk**
- Please help these drafts progress!

Thank you

Questions or comments?

Contact: John Gilmore <gnu@rfc.toad.com>
Seth Schoen <schoen@loyalty.org>

or discuss on the intarea WG mailing list!


Lowest address
~ 16,000,000 addrs
xxxx xxxx
xxxx xxxx
xxxx xxxx
0000 0000
Number of zeroes at right
varies according to netmask.
240/4
268,435,455 addrs
1111 xxxx
xxxx xxxx
xxxx xxxx
xxxx xxxx
0/8
16,777,215 addrs
0000 0000
xxxx xxxx
xxxx xxxx
xxxx xxxx
127/8
net 16,711,680 addrs
0111 1111
xxxx xxxx
xxxx xxxx
xxxx xxxx
Our proposal would require
the second octet to be zero
for loopback.
127.0/16
0111 1111
0000 0000
xxxx xxxx
xxxx xxxx

What about IPv6?

- We support IPv6 adoption.
- “Even if they have deployed IPv6, growing networks must continue to acquire scarce, increasingly expensive IPv4 addresses to interconnect with the rest of the Internet.”

  – *The Hidden Standards War: Economic Factors Affecting IPv6 Deployment*

Death of IPv4 is greatly exaggerated

- IPv4 is the dominant Internet protocol throughout the world
- Unicast traffic is the overwhelming success story of the Internet protocols
- The IPv6 transition is no reason to fail to **maintain IPv4**.
  - “Legacy IPv4 will coexist with IPv6 indefinitely.” – *Hidden Standards War*
  - IETF strategy is to deal with the coexistence of v4 and v6
  - Operational fixes to IPv4 are in scope for IETF and intarea.
- **IETF and intarea did not and should not stop** all future work on the standards that carry the majority of the world's Internet traffic.
- These changes do not compete with IPv6; they help dual stack services.
  - This unicast support takes extremely small changes in OSes
  - Every major deployed service needs to be reachable on both IPv6 and IPv4
  - Relieving v4 scarcity makes that easier, not harder

Broadcast storms?

- One likely motivation for reserving the lowest address for backwards compatibility was to avoid broadcast storms due to disagreements about whether an address was broadcast or unicast
- A layering rule in RFC 1122 already mitigates these, and RFC 2644 discourages honoring subnet broadcasts from outside of the local network.
  - As expected, we were able to trigger **no broadcast storms** when mixing patched and unpatched devices on the same network.