ADDRESSING 101

1. What is in an ***address*** ?

An address is a ***unique "computer-understandable" identifier***. Uniqueness is defined in a domain – outside that domain, to retain uniqueness, one needs to have either a larger address space, or do translation.

An address should ideally be valid regardless of the location of the source, but may change if the destination moves. Fixed size addresses can be processed faster.

Address is fundamental to networking. There is no (non-trivial) network without addresses. Address space size also limits the scalability of networks. In connectionless networks, the most interesting differences revolve around addresses. After all, a connectionless net basically involves putting an address in a packet and sending it hoping it will get to the destination.

Usually we think of layer 2 or layer 3 when we think of addressing. But, even applications ultimately need addresses (which is why transport protocols exist to provide port-numbers). In fact addresses may refer to a node or an interface (latter more common).

==========

2. What is in a ***name*** ?

A name is a ***unique "human-understandable" identifier***.

A name is also ideally something that is ***location-independent*** both with respect to the source and destination. In other words, it will remain ***unchanged if either the source or destination moves***. (Note that in this latter sense alone, IEEE 802 "addresses" can be considered as "names")

In practice, names can be variable length with no limit on length. This means that the name space is infinite, but address spaces are finite. Internet names still have a little bit of complexity and can be pretty long. Lot of work in middleware today is centered around specifying logical names ("the nearest printer on this floor") and worrying about mapping within the infrastructure.

Web names are a big deal because they also represent a brand. This was the basis for a lot of cyber-squatting in the recent past.

==========

3. Why do you need an address?

- Consider a network.. If the network consists just two hosts with ***a point-to-point link, there is no need for addresses***!
- If a ***network consists of more than two hosts***, the source has to know which destination to send the packet to. Immediately we need a unique identifier for destinations.
- Similarly at the lower layer, if a host (or an intermediate node which we call a router) is ***multi-homed***, i.e. it has more than one interface, it faces the problem of deciding which interface to send the packet to. For this you need a unique index (identifier) in a packet, or infer the index from the timing position (eg: in a T-1) line. In the latter case, someone else (application or the higher layers have set up the timing from a knowledge of addresses.

==========

4. Why do you ***need both a name and an address*** besides the ease of understanding?

- When you have more than one address or a name and a set of addresses, you have a ***level of indirection***.
  - Indirection gives you ***flexibility*** because names can be ***organized/administered independent*** of addresses. For example, if a computer moves within an organization (naming domain) but into a different subnet, you can change addresses, but need not change the name. Just change the name-to-address translation.
- Moreover ***putting names in data or signaling packets is inefficient***.
  - This is because names are ***variable length*** usually. Variable length names in headers imply ***unbounded overhead***.
  - It is also complicated for router to look up a name in a routing table because of the ***processing complexity***. Therefore it is convenient to use fixed length identifiers called “addresses”.

    - The telephone network and ATM address may be variable length – but the extra complexity of parsing variable length addresses is done only during signaling and VC/circuit setup. Data/voice sample transfer use either no identifiers (telephony) or shorter fixed length identifiers called labels. The label space is a 32-bit space is managed independently by switches just like frequency space is managed in a cellular network – this leads to excellent scalability in addressing.
- But you don't "need" names.
    - For example, the ***telephone network has only one address, the "phone-number."***
    - You don't dial a "name" on your telephone (except for cryptic names which are directly mapped to numbers on your telephone itself and not hidden from you)!
    - The name-to-address mapping is provided via a parallel directory service. The reason this was done is because of the dumbness of the telephone – it is not intelligent to map names to addresses.

4a) What are the ***other issues about size of addresses*** ?

- A ***large address space allows a large network***, i.e. it is fundamentally required for network scalability
- Address can have structure to encode ***not only the identity of the interface*** (or node as in CLNP, DECnet etc), but also the ***identity of the network*** the interface (node) is on.
    - Intermediate node can ***forward packets to the destination "network"*** instead of the destination "node" or "interface," and the last hop router (on the destination network) uses lower-layer forwarding to reach the destination interface.
    - This is fundamental for scaling the forwarding and routing behavior because the number of forwarding entries for "networks" can be far fewer than those for "nodes" or "interfaces".
- Large address space also makes it ***easier to assign addresses and minimize configuration***. L3 addresses can encode L2 addresses within them if the addresses are long enough (see below)
- Variable length addresses are the most flexible. They save bandwidth in small networks, and don't require committing to the address size. But they make the header difficult to parse (like names), and can have unbounded overhead.

==========

5. What is (address or name) ***resolution***?

- Resolution is the function of ***mapping one address or name to the other***. This problem arises only when there are two addresses or a name and an address that necessitates the mapping.
  - Resolution is a complex and ***fundamental internetworking problem*** in overlay networks (such as IP over ATM) and for multicast.
  - Several techniques are possible for resolution such as snooping (assumes broadcast channel), table-based resolution, closed-form computation-based resolution, and server-based resolution.
- The Internet uses DNS for name resolution and ARP for address resolution. ATM networks also use DNS, but resolve to ATM addresses instead of IP addresses. The telephone network does no resolution because it has only one level of (variable length) addresses

==========

6. What is ***forwarding*** ? Do we need an address for forwarding ?

- Given a packet at a node, ***finding which output port it needs to go to is called "forwarding***" – it is a per-node function, whereas routing may encompass several nodes.
- The ***forwarding function is performed by every node in the network*** including hosts, repeaters, bridges and routers.
- Forwarding **is *trivial in the case of a single-port node*** (or in a dual port node, where the destination is on the port other than the input port) – in this case you don't need even addresses.
  - The earliest bridges used to forward packets from collision domain to collision domain, <u>without filtering</u>. This was ***basic layer-2 forwarding*** or also called "***flooding***". It is still useful and different from the function of repeaters. For ***flooding, you do not require addresses***.
  - For regular forwarding with forwarding tables you need addresses because the address contains a part of the location or topology information of the destination needed for forwarding to work.

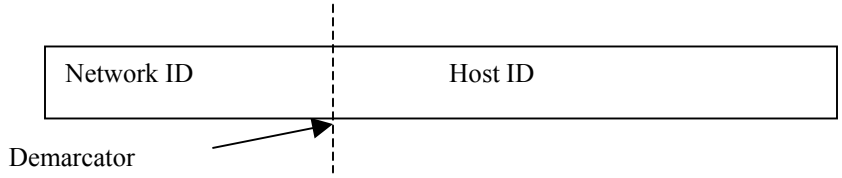
==========

7. Why is there a ***relation between addressing and routing*** ? What is the difference between ***flat and hierarchical addressing*** ?

- If ***you need to scale, there is a relation between addressing and routing***, else there is none.
  - This is one of the differences between the difference between the design of LAN addresses (used by bridges) and IP addresses (used by IP routers).

- ***Routing scales by reducing the size of the virtual network seen by interior nodes***, i.e. reduces the size of routing tables and messages exchanged.
  - For example, consider a distant galaxy. Seen from the earth, it seems like a single node. So a simple strategy is to send messages for planets in the galaxy in that general direction. Within that galaxy, local routes can be found locally.
  - Similarly, in the internet, routers ***forward packets to the destination "network"*** instead of the destination "node" or "interface," and the last hop router (on the destination network) uses lower-layer forwarding to reach the destination interface.
    - In order to support forwarding packets to the "destination network," L3 addresses have structure to encode ***not only the identity of the interface*** (or node as in CLNP, DECnet etc), but also the ***identity of the network*** the interface (node) is on.
    - The network in effect looks like a single virtual node to the intermediate router. This means that the "address size" and "address space" for this virtual network is also smaller – typically subsets of the original address space. And the routers need to maintain forwarding entries only for these networks.
    - This is the essence of "***address aggregation.***" The notions of address aggregation and the procedure of "forwarding to networks instead of nodes" means that number of forwarding entries for "networks" is far fewer than those for "nodes" or "interfaces". This is how L3 routing and forwarding has the potential to scale.

- **<u>Hierarchical addressing:</u>** The simple way to encode a network-ID into a L3 address is to make a prefix of a node address the network-ID.



Hierarchical Structure of a L3 Address

- There is a ***demarcator (either implicitly or explicitly specified)*** that indicates where the network ID ends, and where the host ID begins.
- For address aggregation and scalable routing, two things have to happen:
  a) addresses must be aggregatable (eg: hierarchical), and
  b) the allocated address prefixes must be explicitly aggregated.

- **Aggregatable addresses:**
  - ***Key: addresses cannot be assigned randomly!***
  - For addresses to be "***aggregatable***," nodes in a single network must be assigned addresses with ***the same prefix.***
  - Moreover ***one or more nodes have to be designated as "border nodes"*** which represent the points of entry/exit vis-à-vis the network***.*** Distant routers find directions to reach a border node. Multiple border nodes are used to avoid single points of failure, and to do load-balancing of traffic entering/exiting the network.
  - ***If distant routers "see" multiple border routers,*** they can choose the appropriate border router (that leads to the shortest path) to reach the destination prefix. In this case, we say that the ***"topology aggregation" does not match the "address aggregation,"*** i.e., distant routers have some limited visibility into the topology (actually multiple entry/exit points only) to reach destinations within the aggregated domain.

- **Explicitly aggregating addresses:**
  - Even if nodes within a network have been assigned the same network prefix, the border node should be ***explicitly configured*** to advertise reachability to the prefix, rather than advertising every individual node address.
  - This explicit configuration is done at the border of every level where addresses are aggregated.

- Aggregation does have a cost.
  - ***You lose information*** – but in shortest-distance routing case we don't care. For QoS routing we might care.
  - ***You do lose optimality in routing*** – you tradeoff time/bandwidth resources for scalability.

- ***Traffic intensity at border nodes increases*** – and affects QoS => solution is to have many border nodes and split the reachability between them… This is a ***load-balancing problem*** to be addressed in routing.

- ***Flat addresses do not*** encode the network ID in the L3 address.
  - Flat addressing simplifies address administration (because there is no constraint of numbering nodes with the same prefix) but does not scale (because it requires intermediate nodes to always route to destination nodes)
  - LAN (IEEE 802) addresses are hence flat in terms of routing. They do have a 2-level administrative hierarchy (OUI and the 24-bit number assigned by the OUI owner) – which does not correspond to the hierarchy network physical or logical partitioning.
  - For eg: the nodes assigned addresses 10.* should be reachable through a single border router (and topologically nearby – else routing becomes much less optimal)

==========

8. **Address allocation:** where to place demarcators in the hierarchical address?
   - Recall that hierarchical addresses have a demarcator between the network ID and host ID, which can be "implicitly" or "explicitly" specified.
   - In the early days of IP, ***implicit demarcators*** were used at 8-, 16- or 24-bit boundaries. Such addresses were called ***class A, class B or class C*** addresses respectively (or "***classful***" addresses).
   - The problem with implicit demarcators is that they are inflexible, and lead to inefficient address allocation.
     - **First**, physical networks may be much smaller than the implicit host ID space, and hence would ***underutilize host-ID space***.
     - **Second**, many class-C networks would have to be announced un-aggregated even if they belonged to the same organizational network, and occupied consecutive class-C allocations. This would lead to an explosion of forwarding table entries in the global Internet, ***limiting routing scalability***.
     - The solution is simple: specify ***explicit demarcators!***
     - ***Subnet masking and supernet (CIDR) masking*** were the explicit demarcation techniques used to solve the first and second issues

respectively.

==========

9. Is address aggregation possible only once, or can we aggregate even more at several levels in the Internet ?

Yes, addresses can be aggregated further. With subnet-masks and supernet-masks (CIDR), IP addresses allow multiple levels of aggregation.

- If ***aggregation is desired at several levels*** (subnetwork, network, domain, ISP etc), these levels must be ***hierarchically organized*** (i.e. in a tree manner), and aggregation must be done from the lowest level onward.
  - ***Address allocation*** must also be done hierarchically. I.e. one cannot go to IANA (a central body) to get an address prefix (or address block) -- one needs to go to their ISP which provides their "transit" into the Internet. This is known as ***provider-based address allocation***.
    - For example, top-tier ISP can get a
  - The Internet today largely does provider-based address allocation. But there are two issues which conflict with continued address aggregation in the Internet.
    - ***1. The need for redundancy/availability:***
      - Users typically like to have connectivity to multiple ISPs for simple redundancy and/or load balancing.
      - This breaks the addressing hierarchy because the extra links/connectivity introduces loops ! BGP policies have to deal with this sticky issue.
      - Users may advertise a block of addresses allocated by provider 1 to provider 2, so that they can get redundancy in case provider 1 goes down.
    - ***2. Network migration and renumbering***: If a user network moves from one network to another and does not renumber its network nodes, aggregation cannot be done !
      - This is an irritant to enterprise networks who would like the flexibility to switch ISPs.
      - NAT, DHCP, prefix-lifetimes (eg: in IPv6) help dealing with the renumbering problem (discussed later)

==========

10. What is the problem with static network address boundaries? What is ***subnet and supernet masking***?

- Addresses have a simple 2-part structure: network number and a host number even for arbitrary degrees of hierarchy.
  - This is because any ***intermediate router needs to see only a two-level hierarchy even if there are many levels***, provided the topology is also organized roughly as a hierarchy.
  - The router does not see the lower (or higher) levels of the hierarchy unless it needs to pick one among many border routers.
  - So, we could inform the router on the bit position where the split occurs (for a two-level hierarchy).
- ***Where and how do we demarcate the address bits ?***
  - ***Static (or implicit) demarcation wastes the address space*** (and limits scalability because we have a finite address space), even if we have multiple static classes (eg: the class A, B, C… etc). The ***class structure is also too coarse***.
  - Solution: explicit demarcators.
    - The ***notation 201.10.0.0/21 explicitly informs the router that the split occurs in bit position 21***. The number 21 can be encoded in 5-bits (because the maximum demarcator position is 32)
    - Alternatively, we could have ***32-bit mask that specifies the bits that belong to the network number,*** and the bits that belong to host number. In practice, the network number bits are represented in the mask as a contiguous set of ones starting from the leftmost bit.
    - This is the concept of a "subnet mask" or a "supernet mask" -- essentially serving as explicit demarcators of the network ID within the address.
      - ***Subnet mask is used within an organization (autonomous system) and supernet mask is used between autonomous systems.*** As a result Internet addressing today is "**classless**."
  - An interesting aspect of subnet/supernet masks is that ***masks are NOT carried in data-packet headers***. I.e. they are exchanged only in routing messages, and configured into individual routers.

10a. What is the difference between subnet and VLSM ?

Basic subneting: refers to a fixed demarcation strategy (mask) in addition to natural mask (i.e. class A, B etc). I.e. only a single mask (eg:: 255.255.255.0) can be used for all networks covered by the natural mask.

VLSM: (Variable length subnet mask)
Multiple different masks possible in a single class address space. (255.255.255.0 and 255.255.254.0 could be used to subnet a single class B address space).

- Solves/controls internal fragmentation issues (see below) and makes address space usage more efficient

Address allocation vs Memory allocation:
In general, the problems of address space allocation are similar to that of memory allocation to processes in operating systems. In the latter case, the problems include:

- ***Internal fragmentation*** which refers to wastage within an allocated block (occurs in paging systems)
- ***External fragmentation*** which refers to wastage after blocks have been allocated (occurs in segmentation systems)
  - The solution to external fragmentation is called "***compaction***" which means that the allocations are taken back and reallocated in a contiguous manner. This way, the free space is also contiguous.
- In address allocation, internal fragmentation is solved using VLSMs.
- External fragmentation is solved using CIDR (or supernetting) which allows the allocation of arbitrary address blocks (and prefixes) to networks and enables hierarchical address space allocation.
- Renumbering systems (DHCP) and network address translation (NAT) help in compaction of the address space when there is a high degree of external fragmentation.

==========

11. What is Classless Inter-Domain Routing (CIDR) and supernetting ?

- CIDR was developed to allow the "subnetting" (flexible address space division between network address and host address) idea to be extended to the network part of the IP address too.
  - This effectively would make the addressing "*classless*" for the purposes of routing. Since *inter-domain routing* protocols are the ones that provide routing to the network-part of the IP address, the protocol is called CIDR.

- Supernetting is essentially the idea of explicit demarcation applied to the network ID field -- so that flexible and arbitrary sized address chunks can be allocated to autonomous systems (which can then be internally allocated using subnet masking)

- ***Implications of being "classless": syntax and semantics***

  - In CIDR, we ***prefer the use of the term "prefix" over "network***" because it's more clear that no Class is being implied.
  - ***Notation: 128.221.13.20/20 emphasizing that 20-bits are the network address***. This is also called the <prefix.length> notation.
    - Eg: 198.213.0.0/16 has 16-bit mask shorter than natural 24-bit (class C) mask. The 16-bit prefix is invalid in the class C (natural mask) sense because the class C natural mask has 24-bits.
    - Eg: 198.24.0.0/20 and 198.24.56.0/21 {"***more specific***"} can be aggregated as 198.24.0.0/18 {"***less specific***"}. Note that between each pair of dots, there are 8 bits. /20 means four bits into the third number.
    - The ***masks can be on arbitrary bit boundaries*** and don't have to be on byte boundaries (like the earlier classful boundaries). Note that this still means that networks have to have address spaces that are a power of 2.
  - Note: The dotted-decimal notation is confusing when we try to interpret the CIDR masks with the prefixes ending on non-octet boundaries.
  - ***Longest-prefix matching***: since a routing table can now contain several addresses where part of the prefix would match, the forwarding algorithm has to be modified to match the destination IP address with longest prefix, and not just the first prefix. In other words, it has to match the most-specific prefix.
  -

- ***Implications on inter-domain routing and address allocation:***
  - Pre-CIDR, every class A, class B and class C network needed to be advertised. The number of class B networks and class C networks is huge (64K class B and 2^24 class C networks !!).
    - The key idea is that the CIDR <prefix.length> notation allows all the ***more specific routes handled by an ISP to be summarized into one aggregate***, provided that ***they all refer to non-overlapping subsets of a larger address block.***

    - This means that a provider could get a large address block, i.e. a small prefix (eg: 10 bit prefix rather than a 16-bit class B or 24-bit class C prefix), and allocate smaller blocks to its customers.
    - But ***each of these customer sub-blocks (longer prefixes) need <u>not</u> be advertised***. Only the single small (10-bit) prefix needs to be advertised.
    -
    -
    - ***Customers however don't like provider-based addressing because when they*** move from one ISP to another they need to renumber to remain in the other ISP's CIDR block. This can be a complex and costly administrative task if DHCP or NAT is not available.
    - Renumbering can be done with DHCP automatically. NAT allows avoidance of the renumbering problem by use of private address space.
    - This above allocation procedure of the provider getting large address blocks (small prefixes) from IANA and allocating sub-blocks is called ***provider-based address allocation***.
- In the pre-CIDR era, sites would go to a centralized registry (IANA) to get an address prefix which:
    - A) does not take into account where that site connects to the Internet), and
    - B) allocates only classful prefixes
- The crux of CIDR is that the ***Internet's generally hierarchical topology and administration is now being reflected in the addressing.***
    - To appreciate this fact, observe that in IEEE 802 addresses, the OUI field also implied an administrative hierarchy, but not a topological hierarchy (the IEEE address space is "flat").
    - The ***CIDR-based IP addresses are now overloaded: they have both an administrative significance and a topological significance, and both are expected to be hierarchical !!***
- ***CIDR costs:***
    - CIDR improves scalability, at the ***cost*** of inter-domain routing protocols also carrying subnet masks
    - CIDR also requires a change in the forwarding algorithm. (***need a longest-prefix match*** rather than unique prefix match). The innovative ***data structure used for speeding up lookup is called a "trie"*** which is a tree where intermediate nodes denote part of a prefix!

- CIDR also splits the top level of addresses on a geographic boundary which is not good for providers whose networks might span multiple countries.

==========

12. Other issues in scaling of addresses ? [ADVANCED]

- ***Fixed sized addresses run into a scaling problem because the address space is finite.***
  - Inefficient address allocation adds to this problem: eg: IP's 32-bit address space.

- **Alternative*: large, variable-sized addresses*** (eg: ATM, CLNP have var-sized addresses with maximum length of 20-bytes)
  - This can mean high per-packet overhead, and complex packet processing.
  - ATM tackles the issue of per-packet overhead by using only these large addresses only during signaling, and resorting to small (32-bit) fixed-size labels in the data-path.

- **Alternative*: large, fixed-sized addresses (eg: Ipv6 has 16 byte address,*** IPX has 10-byte addresses)
  - Cost: higher per-packet overhead
  - Gain: additional flexibility for auto-configuration

- **Alternative:** ***multiplexing (share) the costly/scarce address space.***
  - ***DHCP*** is a configuration protocol that has the concept of a ***"lease" (soft state). Soft state or lease*** means that you get an address space for a defined period of time. You can multiplex a set of addresses as long as the user population at any instant does not exceed the set of addresses.
  - The key issue here is that nodes do not "**own**" addresses, but can only "**rent**" or "**lease**" them.

- **Alternative**: ***private-public addresses and NAT:***
  - Use public-addresses for global connectivity and private-addresses for intra-AS connectivity. Private addresses need be unique only within an AS.

- ***Network address translation (NAT)*** maps private addresses to public addresses at the border of an AS on a packet-by-packet basis. This assumes that a public-private address mapping is setup earlier (eg: statically or at the beginning of the flow).
- NAT, like any translation scheme is not perfect partly because of the interdependencies between upper layer protocols like TCP, ftp, ICMP and IP addresses.
- Specifically, IP addresses may be present in transport and application level data

---

13. ***Auto-configuration*** requirements: [ADVANCED]

Configuration revolves around how addresses (primarily) are constructed. Typically end systems only know their hardware (IEEE 802) address…

7 things to (auto-) configure:

1. End systems need Layer 3 address, masks
2. Router finds Layer 3 addresses of end systems
3. Router finds Layer 2 addresses of end systems
4. End systems find a (default) router, name server
5. End nodes on the same LAN discover that they can send directly to each other
6. End systems find the best router for exit traffic
7. End systems communicate on a router-less LAN

Techniques:

1. Manually configure hosts and routers {DECnet}
2. Manually configure routers only {IP, IPv6, IPX, Appletalk (seed router), CLNP}
3. DHCP server {IP, IPv6 (optional)}
4. ARP {IP, Appletalk}
5. IEEE LAN address embedded in host-ID {IPX,CLNP,IPv6 (EUI)}
6. LAN addr = HIORD + L3 addr. HIORD is a fixed extension of the L3-address to form the L2 address {DECnet}
7. Explicit "ES-Hellos" and "IS-Hellos" {CLNP, DECnet}
8. Snoop on RIP traffic for router info {Appletalk, IPX}
9. Best-exit inferred from rcvd traffic {DECnet, Appletalk}
10. Redirects for best-router only (IP, IPv6, IPX)
11. Redirects for best-router and direct end-node (CLNP)

12. Intra-LAN flag for direct end-node (DECnet)

{Needs an understanding of addressing differences in various protocols. Covered in more detail in slide set on comparisons between IP, IPX, CLNP, DECnet, Appletalk etc}

==========

Summary: Addressing requirements:

- Unique
- Address size defines max network size (eg: universal addressibility)
- Usually fixed length in packet headers, but some protocols like CLNP have variable-length addresses.
- Aggregatable: address encodes network ID (aka "locator")
- Allocation flexibility, efficiency and ease
- Auto-configurable (L2 address, L3 network part, host part; addresses of router etc)