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.

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

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

3. Why do you need an unique identifier ?

- 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*** – names can be ***organized/administered independent*** of addresses. 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.
- All said and done, 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 dumb telephone – it cannot map names to addresses.

5. What is ***resolution***?

- Resolution is the function of ***mapping one address or name to the other***. 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*** ?

- 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.
- We note that on a ***directly connected network*** (i.e. using a mesh of point-to-point links or a logical/physical bus (collision domain)), all you need is forwarding -- ***no need for control plane functions to setup a path***.
- Forwarding is a data-plane function which is also called the "critical forwarding path" of every packet.

7. Why do you have ***forwarding tables*** ?

- Unlike in hosts, it is not possible for an intermediate node (bridge or router) to code the output interface number into every destination address to which it can forward. That is, one cannot directly infer (through computation alone) what the output port number is, given an address. This means that ***you need a level of indirection*** at the intermediate node – the forwarding table.
- Note that with multiplexing of the intermediate node and this level of indirection (the forwarding table), we get virtualization i.e. a ***virtual link abstraction across a non-meshed network.***

8. What is ***routing***? How is it different from ***forwarding***?

- Routing is the function of ***finding a path to the destination.***
- Routing is considered to be the ***control-plane complement of forwarding*** (which is in the data-plane or in the critical path of a packet). Specifically, routing ***sets up the forwarding table*** in every node. The size of the routing table (and the information exchanged between nodes to set

up this table) is proportional to the number of nodes (or virtual nodes) seen by the node in the network.

- Routing is also straightforward on a LAN (which can have complex topologies like star, tree, mesh, ring or bus). The ***routing-like control function in a LAN is done by bridges*** which set up spanning trees and learn which interface destination addresses reside.
- Bridges "forward" and "filter" in the data-plane and perform the above routing-like functions in the control plane.
- Only layer-3 devices perform true routing which involves non-directly connected hosts residing in different networks. Which is why routing is commonly seen only in layer-3.

9. Why is there a ***relation between addressing and routing*** ?

- If ***you need to scale, there is a relation***, else there is none.
- ***Routing scales by reducing the size of the virtual network seen by interior nodes***, i.e. reduce the size of routing tables and messages exchanged. This means that the "addresses" for this virtual network are also smaller – typically subsets of the original address space. In the Internet, a router just looks at the network number.
  - Eg: a distant galaxy seems like a node on earth. So you can send messages in that general direction. In the Internet a router may ***consider the entire network 10.* as a single node.***
  - ***Flat addresses give no such clues to routing protocols***. Also, assigning addresses randomly loses advantage that the address space can be aggregated. Now it would be no different than Ethernet… Address administration is important !
  - LAN addresses are 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.
  - Goals of administration: ***Proximity of nodes*** in a single network, address assignment which ***allows aggregation*** in the form of a prefix, and the ***designation of a border node*** is the key to routing scalability. For eg: the nodes assigned addresses 10.* should be reachable through a single border router (and topologically nearby – else routing becomes much less optimal)

- IP addresses or ATM addresses are assigned so that the addressing hierarchy matches the topological partitioning hierarchy. Note that ***splitting address space on geographic or political boundaries may introduce inflexibility to providers*** (like MCI Worldcom who own networks that span these boundaries). Specifically, the addressing hierarchy now does not strictly conform to topological partitioning for routing.
  - Ipv6 allows ***provider-oriented unicast addresses***: there are several levels of administrative hierarchies, but on provider-boundaries (which in turn might refer to topological boundaries). Specifically the split is in the form of registry/provider/subscriber/subnet/interface. The exact position of the splits/lengths is not specified - only guidelines are given. With this a provider may span multiple countries (geographic and political boundaries)
  - Moreover ***in Ipv6, a single interface can have multiple Ipv6 addresses*** unlike Ipv4 where an interface has an unique address.
- 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.

10. What is ***subnetting*** ?

- So, we have to split the address into a network number and a host number for a two level hierarchy or into a series of network numbers and a host number. ***Where do we split the address bits ?***
- ***Static splitting 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***.
- First we observe that any ***intermediate router needs to see only a two-level hierarchy even if there are many levels***. So, we could inform the router on the bit position where the split occurs (for a two-level hierarchy). The ***notation 201.10.0.0/21 tells me that the split occurs in bit position 21***. More generally we could have a subset of bits for the

network number and a subset of bits for the host number. This means that we need a ***detailed bit-mask***, the 1-s' positions denote the bits used for the network number.

    - The latter is the solution used (stored in hosts/routers in forwarding tables, and exchanged in routing messages) though the former notation (/21 for eg) and the corresponding address administration is used/recommended in practice for convenience.
- Retrofitting subneting in the Internet means that ***exterior routers still used the network numbers, but interior routers use subnet masks to expose and exploit additional hierarchy in the host-id address space***. Subnet masks in this case express local aggregation.
- Looking back, the ***original intent*** of the network part of IP address was to identify exactly ***one PHYSICAL network.*** But now, now it refers to exactly ***one VIRTUAL network*** with multiple physical networks inside it. From this angle, we can consider subnet mask as providing a level of indirection -- which unveils more bits of network number within the host id…
    - Note: Currently even exterior routers carry subnet masks and other identifiers (like an AS-number) because of the need to exploit class C address space (CIDR) and introduce additional levels of hierarchy (OSPF areas, BGP domains etc).
- ***Costs of subneting***:
    - ***Routers/hosts need to be changed to know the subnet masks.*** Proxy ARP used to hide hosts unaware of subneting.
    - ***Arbitrary partitioning*** of the address space into network number and host numbers ***could mean that the route tables could be arbitrarily large*** (within the limits) – so you ***need additional management/administration*** to ensure that the "advertised" address space is manageable…

11. Other notes on addressing:

- ***Address/name spaces:***
    - The ***name space is infinite,*** but the ***address space is finite by design.***. This is because of the convention in naming that "." is a delimited, i.e. it separates two parts of a name and you can have infinite number of periods and there is no restriction on the size of the sub-name between

two of these delimiters. Artificial limits due to DNS scaling exist though…

- ***Transport endpoints*** have their own "address space" known ***as "port space".*** The reason this is used is as follows. We expect infinite number of applications to run over IP i.e. multiplexed over IP. ***For demultiplexing, IP needs to identify the apps. IP has only a 8 bit "protocol" field – and can identify only a small set of destinations.*** This protocol field space is strictly administered by IANA, and typically refers to transport protocols.
  - ***So, transport protocols must provide a sufficient address space for demuxing apps***. The solution is to provide ***a locally managed 8-bit port space***, part of which is reserved too! So, if you need more than 256 TCP connections from a single host simultaneously, you will hit a roadblock again !
  - ***ATM NSAP*** addressing provides a ***1-byte selector field*** used like the protocol field in IP to choose the protocol in the end system, but it is explicitly made part of the NSAP.
- ***Multicast*** needs a large address space to accommodate ***dynamic/temporary groups***.
  - **Scoping** is another issue in multicast – you might need to limit the scope/extent of a multicast transmission. The ***Ipv4 addressing does not support scoping***, and in fact multicast support is not mandated in the IP protocol spec. ***Ipv6 corrects these problems***: provides scoping and mandates multicast implementation.
  - ***Aggregating multicast addresses for routing is a non-trivial problem*** especially since multicast sessions are dynamic and span a vast topological extent (larger spatial granularity)
  - ***Inter-domain multicast routing*** is also an area where policy issues cloud technical ones.
- ***Anycast*** address refers to ***any one*** of the interfaces sharing the IP address. Eg: Src-dst communication using specific provider can use an any cast address. Contacting "any" server providing a particular service can use an any cast address. Note that anycast sets up a unicast association where a multicast sets up a multicast association

12. Other issues in scaling of addresses:

- ***Fixed sized addresses run into a scaling problem because the address space is finite.*** Further, inefficient address allocation adds to this problem.
- ***ATM*** has a large ***fixed size NSAP address – 20 bytes*** good enough for scaling purposes, and ***tackles the issue of per-packet overhead by using only fixed-size, 32-bit labels in the data-path.***
- ***Ipv6 has a 16 byte address,*** but uses this in the data path as well – a potential overhead issue on the data path if future (multimedia) apps will use small sized packets. The proponents argue that many networks use a large MTU (Ethernet = 1.5 KB, FDDI = 4 KB, ATM = 8 KB) etc…
- ***CIDR (Classless Inter-domain Routing Protocol)*** restricts address allocation and gives admins ***contiguous pieces of class C space instead of class B space.*** For scaling purposes, the networks having a set of class Cs ***can use one common address for reachability.***
  - CIDR improves scalability, at the ***cost*** of inter-domain routing protocols also carrying subnet masks, and complicating table lookup (***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.
- Other solutions for the scaling problem ***involve multiplexing (!) the (costly/scarce) address space (or subspace). DHCP*** is a configuration protocol which has the concept of a ***"lease" (soft state)*** which 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.
- Address administration (i.e., chopping up arbitrary sized address subspaces for organizations) faces similar problems as ***internal/external fragmentation etc seen in memory/disk allocation seen in operating system.***
- ***Network address translation (NAT)*** allows arbitrary internal IP address assignment as long as the external addresses are unique and follow the external address administration guidelines (eg: pieces of contiguous class Cs etc). This split creates ***"private address spaces" and "public address spaces."*** NAT, like any translation scheme is not perfect partly because of the interdependencies between upper layer protocols like TCP, ftp, ICMP and IP addresses.

- Another solution for the core is to use ***a thin tunneling protocol called MPLS*** which uses a network of MPLS-routers and does not need (in theory) an IP address for every interior node interface. It borrows from the idea of labels in ATM. Similar tunneling solutions are being considered for ***"Virtual private networks" (VPNs)*** which are built on top of the Internet infrastructure, but desire their own private address spaces and restrictions on membership/reachability.