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*** ***<u>independent</u>*** 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
- Large address space also makes it ***easier to assign addresses and minimize configuration***.
- 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).
-

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

- Resolution is the function of ***mapping one address or name to the other***. This problem arises only when there is two addresses or a name and an address which require 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.

9. 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***, 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. reduce 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.
  - This means that the "address size" and "address space" for this virtual network is also smaller – typically subsets of the original address space. This is the essence of "***address aggregation***."
  - In the Internet, a router looks at the network number as the "address" of the "virtual node" i.e. the destination network.. For example: In the Internet a router may ***consider the entire network 10.* as a single node.***
- ***Flat addresses give no such clues to routing protocols***.

- Flat addressing simplifies address administration but does not scale.
- 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.
- For eg: the nodes assigned addresses 10.* should be reachable through a single border router (and topologically nearby – else routing becomes much less optimal)

10. What is address aggregation and what are its implications ?

- Aggregation is essential for creating the illusion that networks are "virtual nodes."
  - The scenario without address aggregation is called "***flat routing***." For example, if an ISP assigns 1000 address blocks (of different types) to customers, and advertizes all these address prefixes to the higher tier ISP, it is said to a flat routed system.
  - For address aggregation, two things have to happen: addresses must be aggregatable, and the allocated address prefixes must be aggregated.
  - The problem is simple with two levels of aggregation (which is what IP first foresaw), but becomes a significant issue with multiple levels
- But for this scheme to work, addresses can't be assigned randomly.
  - Addresses have to be "***aggregatable***," i.e. 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 is the point of entry/exit into the network***.*** Distant routers find directions to reach the 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 ***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. I cannot go to IANA (a central body) to get an address prefix (or address block). I have to go to my ISP.
  - ***The need for redundancy/availability*** breaks this hieararchy.
    - Users typically like to have connectivity to multiple ISPs for simple redundancy and/or load balancing. Provider competition

    can be enhanced if connectivity to multiple ISPs can be maintained by users cheaply.
    - This breaks the hierarchy because the extra links/connectivity introduces loops ! BGP policies have to deal with this sticky issue.
  - ***Network migration, host mobility 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 why renumbering through DHCP is ultimately required by several ISP policies.
    - In IPv6, in spite of a ridiculous number of addresses, prefixes are only temporary and dynamic. This would allow quick migration from ISP to ISP at a host-by-host or even packet-by-packet granularity. This "choice" is expected to foster ISP competition and new economic models.
    - The use of temporary prefixes is also useful for ***mobility***, because the mobile host can get a prefix from a foreign agent (or network) and get its home agent to forward packets to it.
- 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 the problem with static network address boundaries in addresses ? 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***. Alternatively, we could have mask which specifies the bits which belong to the network number and the bits which belong to host number. This is the subnet mask. In practice, the network number bits are represented in the mask as a contiguous set of ones starting from the leftmost bit.
- 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.
- ***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…

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 <u>internal fragmentation</u> issues (see next question) and makes address space usage more efficient

11. Other Issues in address 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.

12. 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.
- ***Implications of being "classless":***
- In CIDR, we ***prefer the use of the term "prefix" over "network***" because it's more clear that no Class is being implied.
- It also uses the ***notation: 128.221.13.20/20 emphasizing that 20-bits are the network address***. This is also called the <prefix.length> notation.
- The prefix may well be 2 bits, i.e. the prefix can be shorter than the natural mask (i.e. what was formerly a network "class"). This is why the term "***supernetting".***
- 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 dotted-decimal notation is confusing when we try to interpret the CIDR masks with the prefixes ending on non-octet boundaries.
- 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.
- Appletalk allows the prefix to end on any number, not a bit value. The CIDR approach is preferred because it facilitates bit-anding in the forwarding path, and also allowing efficient longest-prefix match algos.
- ***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.
- ***<u>Implications on inter-domain routing and address allocation:</u>***
- CIDR enables better forms of aggregation and thus helping to reduce the size of routing tables in Internet cores.
- Earlier, every class A, class B and class C network needed to be advertized. The number of class B networks and class C networks is huge (64K class B and 2^24 class C networks !!). The routing tables can be as huge as that !
- The key idea is that the <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 prefix needs to be advertized. In real terms, the provider would advertize one 10-bit prefix in this case instead of advertising 2^14 small 24-bit prefixes or 2^6 16-bit prefixes – ***several orders of magnitude improvement !!***
- ***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.

13. 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…

- 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.
    - Specifically, IP addresses may be present in transport and application level data

---

<u>14.</u> ***Autoconfiguration*** requirements:

Configuration revolves around how addresses (primarily) are constructed. Are they configured ? Can they be plug-and-play ? Detailed requirements:

Endnode needs to know:

- 1a) Its own layer 3 (network) address
- 1b) Layer 2 address of at least one router (default)
- 1c) Should be able to forward packets to other endnodes on the same LAN without going through a router
    - This should be possible even if the routers are broken
- 1d) Should be able to find the best router (if there are multiple exit routers)

Routers need to know:

- 2a) Layer 3 addresses of neighbor endnodes (for routing advertisements)

- 2b) Layer 2 addresses of neighbor endnodes (for last-hop layer 2 forwarding)

Techniques:

- Pt-to-pt links: End-system "Hello" to find network address. No need for data link address
- LANs: Eg: IEEE 802 addresses allow plug and play because they are globally unique and the source knows its own address (configured upon manufacture). Local IEEE addresses => manager needs to ensure uniqueness
- Other techniques:
  - **ARP:** maps remote network address to remote layer 2 address ***(solves: 1b, 1c,2b)***
  - **RARP:** maps local layer 2 address to local layer 3 address ***(solves: 1a)***
  - **Solicitation/advertisement:** routers/hosts periodically advertize information privy to them. If advertisement interval large, it can be improved through the use of solicitation followed by advertisement (common model: used in Ipv6, mobile IP, Jini etc) ***(solves: 1a, 1b, 1c, 2a, 2b).*** BOOTP/DHCP uses the solicitation model. Hellos sent by hosts or routers are examples of the advertisement model.
  - **ICMP redirect:** tell host which is the better router ***(solves: 1d)***
  - **Random address selection + validation through advertisement** (Appletalk method, also used in Ipv6, and in IP multicast). Ok for small subnets etc
  - **Use MAC address directly as the last few bytes of layer 3 address:** cannot be done in IP. Then only need to get prefixes.
  - **Timeout l3 prefixes and l2 addresses:** request renewal (DHCP, ICMPv6, ARP etc)

Summary: Address requirements:

- Unique
- Address size defines max network size (eg: universal addressibility)
- Location information i.e. topological significance at L3
- Allocation flexibility and ease (eg: hierarchy)
- Autoconfigurable (L2 address, L3 network part, host part; addresses of router etc)
- Aggregatable
- Usually fixed length

- Number of addresses/interface limits network redundancy options. Eg: might need to be connected to one ISP at a time.

Case studies (addressing/configuration):

<u>CLNP</u>:
- End-system hello (ESH) and Intermediate system hello (ISH), redirect messages
- Address lower bits can be filled with IEEE MAC address => with network prefix (from ISH) it can auto-configure
- Uses router cache to store ISH info
- Uses destination cache to transmit directly if necessary. If dest cache miss => transmit to router or all ESs. May get a redirect message.
- Problems: dies when all routers out to lunch ! Redirects to routers contain layer 3 addresses => populate router cache, and fix dest cache

<u>IP:</u>
- IP routing does not route to destination endnode - only to the dest link
  - LAN provides the level 1 routing
- Routers knowing layer 3 address of hosts **<u>done manually</u>** (no ESH)
- ARP for finding layer 2 addresses of neighbors (both for routers and hosts)
- Redirect ICMP messages
- Can talk to other nodes on the same LAN provided the layer 3 addresses have been configured
  - Originally IP endnodes: individually configured. Then RARP to configure L3 addresses.
  - BOOTP and DHCP.
  - ICMP router discovery - too infrequent. Cant be used in practice, and cant be good enough to discover failures of routers

<u>IPX:</u> simplicity and efficiency
- 10 byte adress: 6-byte link
- Solicitation for prefix, router reply + IEEE address => autoconfig solved
- Broadcasts to all routers and gets to know which is best. Multicast would be useful and redirect avoided. But multicast consumes resources in all routers, and also floods the LAN
  - Has the useful property of not sending into a black hole
  - Hello messages to detect liveness can be very slow
  - IPX: flag in data packet. Tell me if you are alive !

Appletalk:

- 3-byte address. Two bytes for network number and one byte for host.
- Use of network number ranges (similar to VLSMs). Can start and end at any number.
- So, you don't do an AND operation like IP on the network number, but look at top 2 bytes and see whether it is within your address range
- Finding a router: snoop on RIP-type messages
- Acquiring L3 address:
  - Pick address at random and send an AARP message hoping NOT to get a reply (collision avoidance)
  - Tries several times and then becomes confident
- Routers get their LAN ranges info from "seed router"

IP, CLNP. IPX can have ***arbitrary levels of hierarchy*** (needed in scaling flexibility). They do not explicitly set aside portions of the address at each level.

- Allows summarization.

==