**Association**

It is a process of joining a network for a purpose of accessing different resources across systems via channels. For that after joining, nodes need to have agreement on a shared view of the communication space, so nodes need to be able to speak the same communication language if they are the part of the same network. Here we have a problem where nodes need to have shared view to communicate, and they need to communicate to agree on the shared view. Solution is made in two steps. First, nodes need to have some kind of prior agreement on way of identifying themselves and channels in the network. Second is that they need to have prior knowledge of a process of finding their own identifier.

There are three **elements of identifier schemes**. First we need to agree on alphabet. We can have letters, numbers and symbols.

Second we need to decide on structure of the scheme. It can be flat or hierarchical. In flat, identifiers do not have some meaningful structure and they can be chosen randomly. Because of that they do not have scalable storage. For every entry in the network we need to have information how to get there on every node. In hierarchical we have organizational structure so we cannot choose any identifier. On the other hand we have more effective storage. Node needs to know nodes in his group and one node form other group. That is much better that all then nodes in whole network.

Third element is semantics. Semantics explains what the meaning of identifier is. It can be name, address and label. Names have functional significance (What?), addresses have topological significance /location (Where?) and labels are semantic-free (port number).

We have option of relating or combining two or more identifier spaces (White pages, DNS, NAT). Composite identifier schemes are used for functions that communicate through multiple levels (Postal service, SMTP over IP).

**Identifier acquisition mechanisms** were in the early days done manually. Configuration of IP address was normal and it is still done in practice, but lot less. Reason for that are mechanisms for automatically getting identifier. 3 such mechanisms are Dynamic Host Configuration Protocol, Stateless Auto-configuration (SAA) and Auto-association.

In **DHCP** server has a database with all the configuration profiles and options for clients. When client wants to associate, he broadcasts the discover message. If there is some DHCP there, he will replay with configuration options for the client. Then client needs to choose server if there was more than one answer and send request with configuration parameters to server. Then server needs to acknowledge the connection by sending the message that usually has default gateway, address and domain name. After that, client is connected. When he decides to disconnect, he sends release message to server. If he crashes before sending that message, server waits fixed timeout and then releases that connection spot.

Good thing about DHCP is that we configure one server and we can have many automatically configured hosts. We can also reuse same addresses for many hosts but we have max numbers of hosts that can be connected at the same time. There is access control over mac addresses. Bad side is that we need to configure server which can be time consuming depending of the size of network. Another disadvantage is that server is single point of failure of the whole network. In practice it is usually solved with backup DHCP server.

**SAA** or IPv6 Stateless Address Auto-configuration is a process of association on IPv6 networks. Here we do not keep states like in DHCP. Every node, when tries to connect, broadcasts some random number that he chooses, usually mac address, to all its new neighbors. If he doesn’t get response, his number is unique and he uses it as lower 64 bits part of the address. Then node sends solicit message to the router. Router returns unique upper part of the address for that part of the network. Now node has both parts of the address and he is associated.

Good thing about this process is that every node can find its own address, without need of the server. Bad thing is that there is no possibility for fixed configuration and there is no access control.

//some things are missing here but I really can’t understand what Manolis is saying.

**Auto association** is used in WIFI and GSM. In first phase, called scanning, node listens on all frequency channels for periodic BEACONs of available Access Points. Then it selects an access point with highest-signal-strength, or some other criterion. In the next phase it exchanges authentication information with AP. If authentication is successful node sends ASSOSIATION\_REQ and gets rest of the information needed to establish full connection from the AP. There is a case where AP doesn’t send BEACON messages. If node knows that there is AP it can broadcasts PROBE\_REQ on all possible frequency channels to find that AP. Rest of the process is the same.

More complex service environment requires more identifier schemes. For example overlay networks, mobility, and multi-homing. If we for example didn’t have such function in GSM networks, we would need to change phone number every time we change the phone. Another example is session that needs to be kept alive even if we change location in network or for some long term connection or some time-consuming data stream. We have three solutions here that resolve this problem.

**HIP** or Host Identity Protocol practically inserts new layer between IP and transfer layer. Generated host ID can be used from a different IP addresses. Security of session is accomplished by using Diffie-Hellman key-exchange where two sides both have their public and private key. In combination of using them they can privately compute same number that they use to authenticate communication. In HIP, when some node initiates communication, responder sends pre-computed package that contains D-H key and some puzzle, back to the node. This puzzle is used as a defense from denial of service attacks, because node needs to solve the puzzle if he wants responder to start computation of D-H. So, node needs to solve the puzzle and to compute D-H using his private and public key, and only then to send it all back to responder. If all is in order, responder computes D-H and sends back message of acknowledge to the node. If node or responder change location, they need to update IP-host ID binding with new IP. That is done with a three-way-handshake. After first phase of HIP, both nodes are equal, and moved node sends update location message to the responder with new location, sequence number for the session and D-H key. Responder then updates binding of new location and sends acknowledge response with location, sequence number for the session and D-H key. Final step is for the node to send confirmation of update request once again, and location is updated.

**LISP**, or Locator ID Separation Protocol, divides use of IP address identifiers as two isolated identifier spaces. One is used for routing, and other one for session. At edge-network ID space we use end-point IP address (EID), and at core-network we use route locator IP address (RLOC). Border-routers encapsulate packets traveling towards the Internet core. At Ingress Tunnel Routers EID-to-RLOC mapping is performed. The Ingress Tunnel Routers adds an IP header that uses RLOCs. Border routers de-capsulate packets coming from the core network. When packets reach the Egress Tunnel Routers, the outer header is removed. End-point identifiers (EIDs) are used in inner header. Route locators (RLOCs) are used in outer header.

**SHIM6**: Site Multi-homing by IPv6 Intermediation is used for multi-homing in IPv6. Goal is to identify a group of IP addresses by a group-ID and use it as communication endpoint. The first address used to setup a communication is used as the session ID as long as communication lasts. If any other address from the group is used, the session IDs is preserved. This process is done only by agreement of two sides. There is no need for additional functionality.