Mobility Made Simple

by
John Day
Eduard Grasa

March 2016

1. Introduction

In these notes, we have taken a very brief look at issues in various topics: a) in protocol design, b) the nature of layers, and c) naming and addressing. We are working toward an understanding of how mobility should work in a well-formed, complete network architecture.

In protocol design, we created layers using what seemed to be a logical separation of functionality, which turned out to split a dependency across the layer boundary, i.e. not to follow the inherent structure of the problem (its invariances). This led to unforeseen consequences that just got worse with successive attempts to work around them. On the other hand, with RINA, doing what the problem says kept the design simple and it just worked without any patches or special cases. (There is much more that could be said about this.)

Translating what we learned about protocols to layers, we found that one can't just draw a boundary anywhere. Not only do some functions not work cleanly (fragmentation), but layer separation had to be compromised. But again, if one follows the invariances inherent in the problem, the solutions remain simple. Even better, we found that the functionality of layers for managing resource allocation were the same, just applied to different ranges of the problem space. Creating the possibility of using divide-and-conquer to conquer scaling issues![1]

Lastly, we looked at the nature of naming and addressing, where again, doing what the problem said produced a simple solution with the added benefit of reducing router table size by a significant factor. At first this was a surprising result. But another one of those that once you think about it, isn't surprising at all! As we have argued, none of this is really new. At least the indications, if not the facts, have been known for 40+ years. Even the organization of the functions isn't that different, primarily we have cleaned away the kind of kruft, interdependencies, inconsistencies, and redundancy that can accumulate when protocols are developed independent of each other.

We now have most of what is needed to look at mobility, which as we said at the end of the note on Naming and Addressing is actually quite simple.

2. So What is the Problem?!

[1] Grammatical redundancy intended.

What is it that makes mobility appear to be so hard? Well, obviously because one or both of the applications are moving – their location is changing! This brings with it what would seem to be some major problems. Most relate to addressing, since addresses, at least in the vernacular, are used to locate places, and places don't move.

For a very long time on the IETF's End-to-End list, every few months someone would ask, "Why can't I take my IP address with me when I move!?" (You would think that the fact it was called an *address* would be sufficient to answer the question.)[2] One answer was, 'you can, it just wouldn't be an address." Of course, part of the problem was that they had the example of the so-called MAC address that was assigned at the factory. Knowing that was what most people had in mind, I began to answer the query with "What was the address of where you were born? I want to send you a letter." to which I would get, "But I don't live there anymore." What!? You didn't take your address with you when you moved!!? Why not!? This generally got the point across. An address is location-dependent relative to a network, whether streets or computers.

Why did it work for MAC addresses? Ethernet was a very special case. In its original incarnation as a multi-access media, when a PDU is sent over an Ethernet segment, every station on the cable sees the PDU. When a station detects a PDU with the same MAC address as that assigned to it, the station copies the PDU off the wire.[3] The only property a MAC address requires is uniqueness in its own Ethernet and it doesn't even require that (see footnote). If MAC addresses were used for routing, a forwarding table entry would be required for each MAC address on the network. Since routing on Ethernet was either broadcast or of very limited scope, the forwarding tables were always small. Strictly speaking, they were not communication addresses because they were not location-dependent relative to the graph of the network.

IP addresses had this characteristic before CIDR. They were assigned in order and also named the interface. But IP networks don't work like Ethernets. All packets do not pass all hosts. They have to be routed. Unlike letters, one could not tell from the address that two packets were going to places near each other.[4] Consequently, virtually every new block of IP addresses assigned required another router table entry. By the end of the 80s, router table size became an issue. It was far past the time that IP addresses had to really be *addresses*.

What *is* the problem with mobility?

[2] Apparently, it still hasn't sunk in. In 2016, the IETF is discussing the need for permanent IPv6 addresses!

[3] Note that because of how it works, a multicast address in Ethernet is an *ambiguous* address, several stations with the same address; rather than the more formal (and useful) definition as the name of a set of addresses that resolves to all members of the set. The Ethernet definition works in a multi-access media, but implies flooding if used in a wired media with relays. (Multicast spanning trees use the set definition whether implicitly or explicitly.

[4] A postal address indicates which letters are going in the same direction, or to the same country, or the same city, etc. IP addresses didn't.

Clearly, the problem is that the entity assigned an address is in a container (device) that is moving! As it moves its points of attachment to the network are changing, so where it is relative to the graph[5] of the network is changing. If its address is to indicate where it is, then it is clear the address is going to have to change. If addresses aren't changed then there will have to be an additional entry in the routing table for every address associated with a mobile device. So what properties do we need for mobility?

1) Address(es) that are used for routing or for points of attachment need to be able to change as the device moves.
2) A short enough time interval to detect a new point of attachment, acquire it and be able to transfer data on it, ideally before losing an old one; and
3) The time to propagate knowledge of the new address within the layer to the routing information of members of that layer should be small relative to the time interval between address changes. In other words, the layer should have sufficient time to reach a new stable operating state.
4) All of this should be done without the loss of traffic associated with the flows.
5) The applications communicating and the device should have names that are invariant with respect to changes in location in the network, so that they can be managed and remain accountable, and so that authentication is location-independent.
6) Solutions must scale to very large numbers of devices, sometimes moving quite fast.

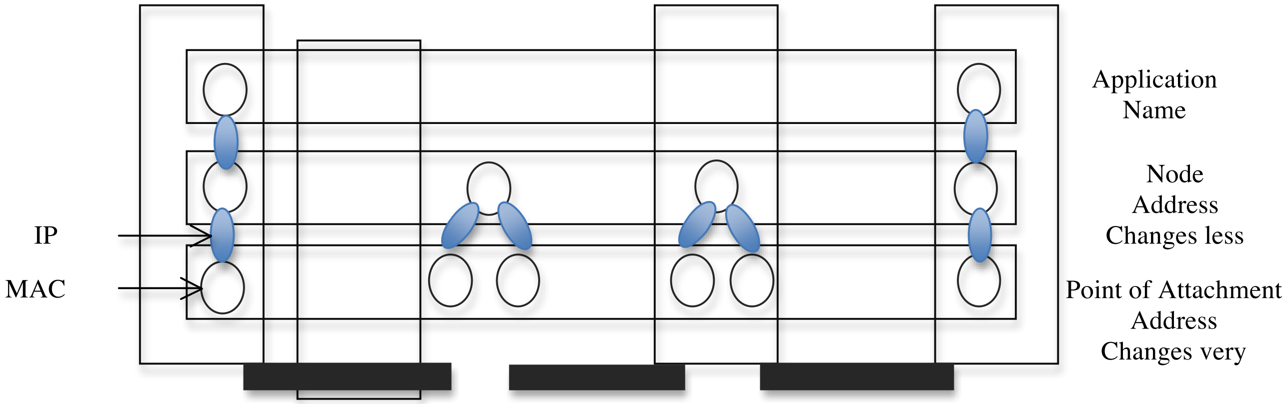

Figure 1. On the right are the identifiers a complete architecture requires and an indication of the frequency with which they change for a Mobile Host (MH). The Internet has two point of attachment addresses (left) for each point of attachment, both have to change at the same rate. However, something has to remain unchanged. This is the problem MIPv4, MIPv6 and PMIPv6 try

3. Why Does the Internet Do Mobility the Way That it Does?

If we look at the above requirements and consider what we found in the note on Naming and Addressing, it should be pretty clear that the problems mobility presents to the Internet architecture are pretty severe. We need an identifier that does not change as the host moves and one that does. In fact, it would be really nice to have two kinds of identifiers that change when a mobile host moves: one kind, for the point of attachment

[5] Note that the use of the term "graph." Because that is what it is. We do not use the term "topology," because it isn't one. Nor are probably 99% of the uses of that word in these discussions.

address that changes frequently (and there could be more than one), and one for the node address that changes less frequently, and then one that doesn't change at all for the application name.

The problem is that the only identifier assigned to an entity in a Mobile Host (MH) that has scope greater than the system itself is the IP address. But that is the identifier that is supposed to be the point of attachment address. As we saw, it can be location-dependent relative to the graph of the network, but it is also route-dependent and is the identifier that changes most frequently!

There is no node address and there is no application name in IP. Domain names are macros for IP addresses. Domain names are not used for routing, and do not satisfy the properties in the list above as changes to the DNS name do not (and cannot in general) propagate to users of an IP address if the DNS mapping is changed to refer to a different IP address. Well-known ports are local identifiers, which with the IP-address, name a path to an arbitrary instance of an application. Just what we don't need! The path is changing! Consequently, the IP address can't change but where it is must, so the routing infrastructure must be cognizant of IP addresses that are mobile. If nothing is done, then every router that might see a MH address would have to have a separate routing table entry for that address, since it wouldn't be aggregateable once the MH leaves the home area where the address was originally assigned. In addition, the routing updates would have to be often enough and propagated through the entire IP routing infrastructure faster than the rate of change of points of attachment. Basically, this would be equivalent to throwing out CIDR and instead of allocating blocks of IP addresses randomly, it would be allocating individual IP addresses randomly. In other words, take the size of today's routing tables (pushing 500k) and add the number of mobile phones, tablets, IoT devices, etc., i.e. router tables of several billion and growing very fast. That isn't going to work.

There are 3 primary approaches to doing IP Mobility: Mobile IPv4 (MIPv4), Mobile IPv6 (MIPv6), and Proxy Mobile IPv6 MIPv6). They are all based on a common theme. Let me describe MIPv4 and then we can consider the 'improvements' for v6. (All of you know this so bear with me.)

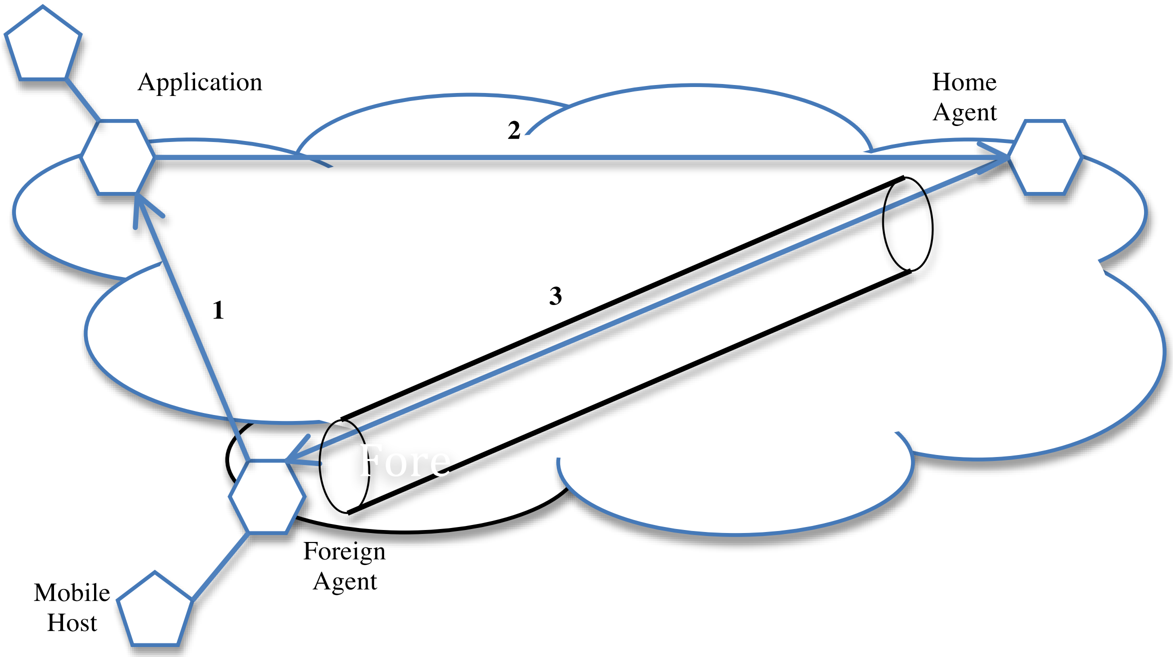

Figure 2. With MIPv4, when the Mobile Host (MH) attaches to the Foreign Agent (FA), it contacts its Home Agent (HA) to create a tunnel. When the MH opens a connection 1) to the Application. The Application responds 2) sending the PDU to the MH's normal IP address. The HA intercepts it knowing the MH is elsewhere 3) forwards the PDU through the tunnel to the FA who delivers it to the MH.

3.1 Mobile IPv4 (MIPv4)

The approach taken by MIPv4 (see Figure 2) is that ultimately the IP address does not change, but IP tunnels are created to deliver the PDUs to where Mobile Host (MH) is attached to the network. Mobile IPv4 (simplifying a bit) designates the router the MH is nominally connected to as its Home Agent (HA). When the MH moves it is attached to a different router, this is its Foreign Agent (FA). The first thing the MH does is register with the FA, which creates a tunnel back to the HA. Any PDUs sent to the MH will be intercepted by the HA and put into the tunnel, routed to the FA, which will deliver them on the interface the MH is connected to. (See Figure 2).

To avoid the overhead of having all traffic flow to the HA, then to the FA through the tunnel, an MH can (if and only if the FA allows originating packets from addresses it does not own) use "triangle routing". When performing triangle routing, the MH opens a connection to some Application (App), sending it directly to the application's IP address but using its HA IP address as the sending address of its packets. The App responds using the MH's IP address which goes to the HA and is forwarded through the tunnel to the FA, which delivers it to the MH. This clearly creates asymmetric transit times and skews round-trip-time estimates. This works if the Application is also mobile.

So the first 'enhancement' to be made is to bring the router the App's Host is connected to into this charade. The FA creates a tunnel with the App's router, lets call it a Destination Agent (DA). Now the MH generates traffic that goes from the FA to the DA directly and the return traffic goes from the DA to the FA in a tunnel, but does not involve the HA. This was optional in MIPv4.

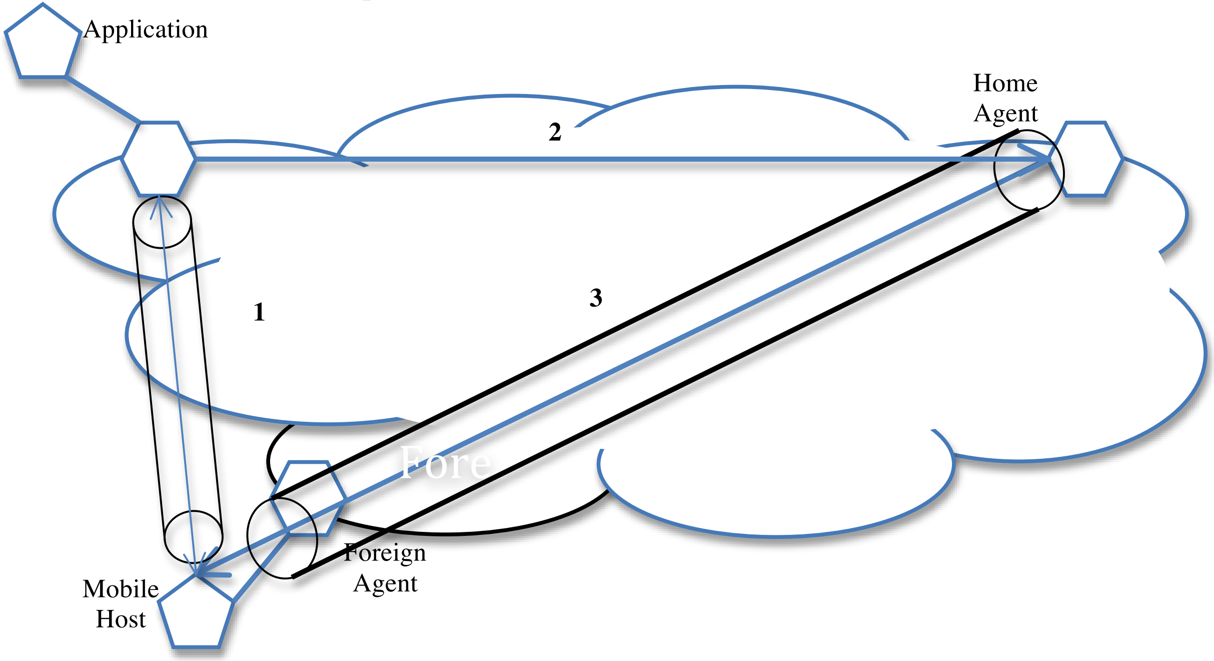

Figure 3. MIPv6 works very much like MIPv4, except the tunnels terminate on the Mobile Host, rather than the Foreign Agent, and to avoid triangle routing the tunnel between the Application and the Mobile Host is mandatory rather than optional. The tunnel with the Home Agent then is used primarily for setting up connections.

3.2 Mobile IPv6 (MIPv6)

The main differences between MIPv4 and MIPv6 (see Figure 3) are that the 'enhancement' is no longer optional. Two tunnels are required to the HA and to the DA. The ends of the tunnels are now in the MH. The MH registers on a Foreign Link and is assigned a "Care-of" address, which is the end of its tunnels. This means that the "Foreign routers" do not need to be aware that the MH is mobile. In addition, considerable attention is paid to security requiring authentication both at the HA and DA. (This creates more overhead.)

3.3 Proxy Mobile IPv6 (PMIPv6)

The rationale for PMIPv6 is based on three factors: 1) MIPv6 requires changes to the protocol stack in OS kernels and is hard to deploy, 2) the hand-off is not efficient and incurs large delay, and 3) kernel modifications opens the opportunity for security attacks. Basically, PMIPv6 goes back to using a FA, and is limited to "*localized networks* with *limited topology* where hand-off signaling delays are minimal." [Egli, 2014] Even granting that the author meant "graph" when he wrote "topology," one can only guess what a "limited graph" is. (?) However, one would also conclude from this restriction that the hand-off isn't much more efficient than the hand-off in MIPv6.

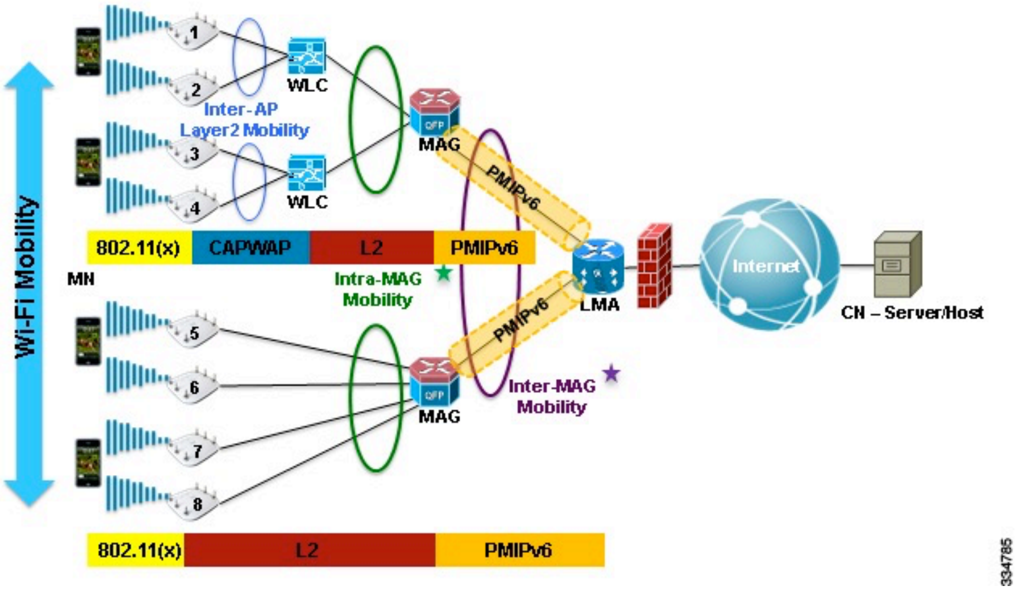

Figure 4. Cisco's view of what PMIPv6 looks like. Notice that there are two levels of hand-off within the WiFi Network and across different WiFi networks. This will lower the frequency of tunnel changes.

PMIPv6 defines several 'new' terms:

> **Local Mobility Domain (LMD):** 'A network that is PMIP enabled and consists of 1 LMA and multiple MAGs.' This must be that "subnet with the limited topology".
> **Local Mobility Anchor (LMA):** 'All traffic from and to the mobile node is routed through the LMA. The LMA maintains a set of Routes for each MH connected to the LMD.' This is a proxy-HA and a single point of failure all rolled into one.
> **Mobile Access Gateway (MAG):** The MAG performs the mobility related signaling on behalf of the MHs attached to the access links and is usually the first hop router for the MH.' In other words, more or less a Foreign Agent.
> **Mobile Node (MN):** a Mobile Host (MH). I am sure there was a good reason for not re-using the same term.[6]

Now things get interesting. There is also:

> **Proxy "Care-of" Address (Proxy-CoA):** 'the IP address of a public interface (no private addresses) of the MAG, the end of the tunnel to the MAG.'
> **Mobile Node Identifier:** 'A unique identifier of the mobile node, usually a MAC address.' Of course MAC addresses constitute a security problem by themselves, as they enable tracking of individual MHs by bad actors (like advertisers…).
> **Home Network Prefix:** 'Prefix assigned by the LMA to the MH.' The prefix is the longest prefix from the point of view of the LMA.

[6] Like Internet of Things: I thought it had been an Internet of things from the start.

According to Egli, the primary environment for PMIPv6 is a campus WiFi network, where the MH moves between WiFi Base Stations - which connect to different routers (MAGs) - and the LMA is the border router to the campus network (see below). PMIP turns things around and makes the LMA the HA, so the rest of the Internet sees the LMA as on the path to the MH. The advantage here is that rather than have tunnels potentially spanning the Internet, the tunnels are localized to the subnet that the LMA and MH are on at the cost of creating a single point of failure. *PMIPv6 is essentially a NAT for mobility.* There are two additional procedures for "fast hand-off," a predictive one, where the next MAG is known and a reactive one where the next MAG is not known. There is no hand-off between LMAs, which greatly limits its applicability. Of course there are the usual security procedures to be considered when setting up the tunnels and new MAGs interacting with the LMA.

3.4 IETF Distributed Mobility Management (DMM) Working Group

PMIPv6 relies on a single, centralized mobility anchor (the LMA), creating a bottleneck and a single point of failure and forcing all the traffic to/from Mobile Nodes in a mobility domain to go through the LMA. "DMM protocols being considered at the IETF aim at distributing mobile Internet traffic in an optimal way while not relaying on centrally deployed mobility anchors" [Lee 2013]. The model under consideration is still a flat (single layer) Internet architecture with all-IP backhaul support, where multiple mobility anchors are distributed among the access network.

[Giust 2015] provides a discussion on how to materialize the DMM goals in practice using three different approaches:

- *Using an evolution of PMIPv6* – therefore tunnels - distributing the forwarding amongst multiple mobility anchors but keeping a centralized database;
- *Using SDN* - even more centralized, a central controller has to configure the forwarding tables of all mobility anchor routers, also requiring these mobility anchor routers to do translation of IP addresses in the packets from/to mobile nodes;
- *Using BGP routing and DNS*. This solution uses no mobility anchors at all, just routing updates to update the location of the Mobile Node in the network. However, two problems with this approach are: i) since the scope of the network layer is the whole world routing convergence is too slow;[7] and ii) since the Mobile Node IP address does not change (it is not an address), routing overhead is high and contributes to further increases in the size of forwarding tables tracking mobile nodes.

3.5 Analysis

This is an excellent example of how complicated the problem becomes with an incomplete architecture. The conflicting requirements of needing a global identifier that *doesn't* change and that the only global identifier one has *must* change leads to the use of tunnels to get around the problem. Of course, setting up and tearing down tunnels is time-consuming and fraught with errors. Clearly, every time the MH changes points of attachment all of these tunnels have to move. The fact that these are the points of

[7] Currently, it takes 20 minutes for BGP to stabilize. Needless to say one could need to update the location of a mobile node several times in 20 minutes.

attachment makes the problem worse. In addition for some period of time there must be a new tunnel between the old FA (MAG) and the new FA (MAG). For MIPv4 and MIPv6, the old HA to FA (MH) tunnels can be terminated as soon as a new HA to FA tunnel is created. It will take longer to create the DA to new FA tunnels before the old FA to new FA tunnel can be terminated. For MIPv4, or 6, to optimize the routing or even just to make the traffic "better behaved" requires the Server to incur additional overhead of supporting tunnels for the MHs, that change as frequently as HA tunnels do. One could easily see Server owners partitioning MH users from normal users. Of course scaling is a major problem here. There is considerable overhead on all concerned. Everything is a special case.

PMIPv6 presents some special problems. As noted above, it is basically a NAT/Firewall approach with the LMA the "NAT/Firewall." The description of PMIPv6 indicates that it is intended for use with campus WiFi networks. (We will consider cellular in a bit.) When one joins a campus network, an IP address is assigned. 802.11 already handles hand-off between base stations. So either PMIPv6 is unnecessary, or one might imagine that a campus would use 802.11 hand-off between Access Points in the same building and would use PMIPv6 hand-off between buildings. OTOH, fewer "parts" or protocols is always better. This is made more absurd by the fact that PMIPv6 uses the same identifiers, i.e. MAC Addresses, to do the hand-off as 802.11 does. Any scenario that proposes a two-tier hand-off scheme (which is not a bad idea) with PMIPv6 as the 2nd tier would seem to run afoul of the assumption that PMIPv6 is intended for so-called "limited-topologies."

PMIPv6 is also used by LTE to hand-off calls between 3GPP and non-3GPP access networks (such as between LTE and a local WiFi network). Management is established at the level of the LMAs (usually co-located with LTE Packet Gateways), so that the session can be handed-off from LTE to the WiFi network and vice versa. This would seem to stretch the "limited topology" constraint even more.

How well do these scheme meet the 5 requirements?
Lets see:

1) *Addresses locate the MH*. The addresses of one end of two tunnels (FA and the DA) indicate where the MH is. A layer management protocol is required for FA to HA negotiation and for FA to DA negotiation. If the server environment does not support MIPv6, then the connection will have to resort to triangle routing. This is known to create problems, besides being inefficient. This problem of course does not exist with PMIPv6, but PMIPv6 does create a potential bottleneck and single point of failure.
2) *Responsiveness to location changes.* The time to make changes would be hard to calculate. The MH first has to acquire the new point of attachment, then make contact with a new FA, probably be authenticated (which would require more than one round trip) and create a new tunnel with the HA. Once this is done, the HA could use the new FA tunnel (this is only for new contacts). The MH would also have to notify the old FA and all of the DAs and create tunnels with them.

The old FA and the DAs would have to wait 5 seconds (Max TTL) before closing the old tunnels to ensure there were no old packets in route as would the MH.[8] The existence of PMIPv6 indicates that this is too slow and too cumbersome. At the same time, as described above, it would seem that because 802.11 does hand-offs that the environments for PMIPv6 would have the same problems of being too slow and complex. There should be an advantage that the tunnels stay within a subnet near the MH.

3) *Not lose traffic.* In theory, it shouldn't lose traffic. However, with tunnels there is always the opportunity for data to be lost. There will undoubtedly be more out-of-order PDUs and probably some delivered quite late. It is unclear how congestion control will affect this. However, given that the tunnels are direct (triangle routing is eliminated) it shouldn’t be too bad. The real question is can all of the new tunnels be set up before the attachment to the old FA is lost? That could be a fairly tight bound. With PMIPv6, there is the potential for the LMA to be a bottleneck, this could increase the frequency of congestion responses, slowing throughput and increasing discarded PDUs.
4) *Manageability.* The MH retains its IP address, so there is always the means to contact an application through the combination IP address and port numbers, although the initial path will be through the HA and may require new tunnels be created. (It is unclear how SNMP works for example. Does it use the HA-FA tunnel or is it given a new one? I presume the latter.)
5) *Scales.* It is hard to believe that any of these will scale. All initial contacts will have to be authenticated (time consuming), creating tunnels is always a significant complication, round trip times are likely to be near the Internet average. But with a considerable increase in management traffic to set up and tear down tunnels and no effect on router table size.

The fundamental reason it is so complex is the lack of a full addressing complement. There is nothing to hold on to but the IP address. But the real problem with all of these is that are point “solutions.” They require considerable additional mechanisms and protocols, which require additional security mechanisms, and have no other benefits.

Can we do better? Definitely.

4. Mobility in a Well-Formed Architecture

If we refer back to the note on Naming and Addressing, we can see immediately how mobility should work and the answers to our questions.

How does it work? Mobility is nothing more than multihoming where the points of attachment change a bit more frequently. But points of attachment fail on occasion anyway and must be recovered either when the point of attachment is restored or when new ones are available (which is the same thing). This may be a node in the same (N-1)-layer or the system may join a different (N-1)-layer (enrollment). Other than perhaps

[8] This might be shortened if they can determine the TTL being used by both the DA and MH on a given tunnel.

different policies to accommodate the increased rate of change, nothing new is required over and above what is needed to provide multihoming. Even a different policy would not necessarily be special. One can easily imagine a policy that increased or decreased its sensitivity to changing points of attachment based on either past behavior or the rate of change of GPS coordinates. For example, if I am working at home all day why should my cellphone be continually looking for new points of attachment? They aren't going to change. But if I go out run errands, the points of attachment would be changing more frequently and the policy would want to be more active. This could easily be the default policy. But with a complete naming architecture, there are No special protocols required; No Foreign Agents; No Home Agents; No Anchors; and No Tunnels.

Notice also that this is not a point solution. We have also solved multihoming, reduced router table size, and made multicast much simpler.

What about the questions? The devil is always in the details!
Yes, but only if it is done wrong! Lets look at the questions:

1) *Addresses locate the MH.* The short answer is that the address of *any* node in the (N)-layer locates that node within the graph of the (N)-layer. The (N)-layer may be the application layer, or any network layer beneath it. To understand how requires a bit more detail in how things should be defined. First, we recognize that each node has logic that executes all of the data transfer and layer management functions of this node in a layer. This activity can be modeled as a process with multiple threads or sub-tasks, regardless of how it is actually implemented (all of these sub-tasks have common access to the state information of the node, called the Resource Information Base or RIB.) Every application process is given an application process name, and since the communicating process within a node is a process (called an IPC Process or IPCP)[9] it has an application process name just like any other[10] and this name can be used to communicate with it. However, within the (N)-layer, an IPCP is assigned a synonym to the IPCP-instance whose scope is limited to the (N)-layer and may be structured to facilitate its use within the (N)-layer, i.e. the synonym is structured to reflect its location relative to the graph of the (N)-layer. This synonym is often called an *address*.

   If the MH, which contains this node, moves too far the address will no longer be aggregateable causing an increase in router table size and potentially less efficient routing. This means that its address within the layer must change to keep router table size manageable and efficient. This is actually quite simple. A new synonym is assigned to the IPCP, which reflects its new position in the graph. This new address is used in the source address of all connections originating at this IPCP, thus notifying the other end of the change, which it now uses as the

[9] OSI defined this as a (N)-subsystem – the intersection of a layer and a system. ISO 7498-1

[10] The distinction between user processes and system processes has always been an artificial as is the distinction between hardware and software.

address in return traffic.[11] The routing updates cease advertising the old address and only advertise the new address. The old address dies off and the new address is in place. Note that the application name of the IPCPs and of the application itself never change.

2) *Responsiveness to location change.* Most readers now will be thinking, ‘a routing update takes far too long to be responsive enough for this environment!’ This is true in traditional static network architectures, where there are basically two scopes: the very small point-to-point data link layer and the whole world of the network layer.

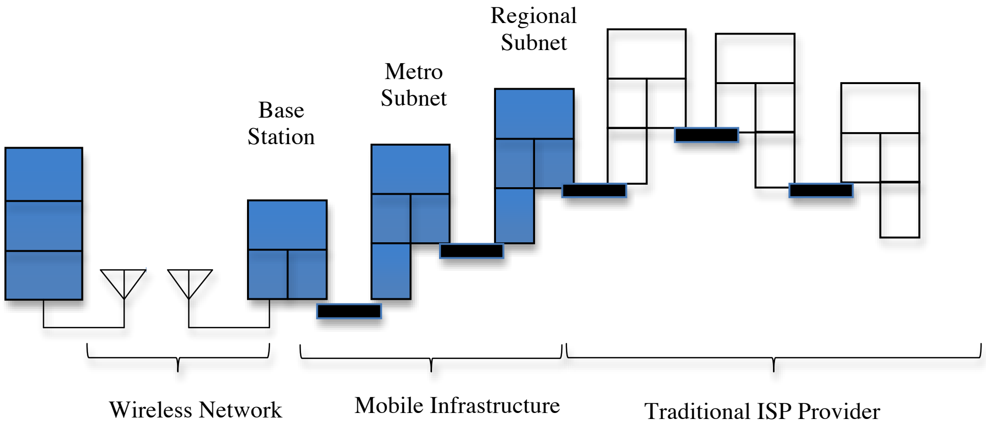

Figure 5. By using the repeating structure, the network can be designed to accommodate the rate of change of points of attachment simply and easily. (Assume there is a network of routers between border router, space does not

But recognizing that all layers do the same functions but for different ranges of bandwidth, QoS, and scale means that the number of layers is not an architectural issue but a network engineering issue. By creating more layers of the same rank or layers of greater rank, the size of routing tables and the time to update them can be bounded by design. This would imply that the lower layers where points of attachment would change frequently would have small scopes that could be updated quickly; higher layers would have greater scopes where points of attachment changed less frequently and updates would take longer but still a fraction of the time that a MH moving across it would change points of attachment, and so on!

Just as in the case of wireline networks, the repeating structure can be used to improve the management of bandwidth, increasing traffic density, and managing router table size, here the number of layers can be used to handle any density of MHs and virtually any rate of change. (In this description, we assume a typical cellular network, where MHs are one hop from base stations that are connected to a wireline infrastructure. The same technique can be used in ad hoc or so-called

[11] Since these layers are securable containers, including that all members of the layer are authenticated when they join and there is no requirement for any (N)-PCI to be delivered to the lower layer in the clear, there is little or no chance for spoofing. This could be done as described here or by a layer management PDU sent by the Flow Allocator.

mesh networks,[12] but there is more engineering to be done to match the structure to the nature of the network.) The depth of the layers could be different in different parts of the same network depending on the requirements. Again it is worth noting:

The Number of Layers is not fixed by the architecture, but by the requirements of the specific network and may be dynamic.

Furthermore, the number of layers could change easily as the demand on the network evolved.

But more layers means more overhead! Maybe a little bit, but not much. First of all, by using the same protocol for data transfer in each layer and since the protocol is defined to be invariant with respect to syntax, the fields of the PCI can be chosen to fit the scope of the layer. For example, for layers of small scope address fields (the main source of overhead) might be 8 or 12 bits. Depending on the RTT of the layer sequence numbers could be smaller as well. Higher layers would have larger fields.[13] Furthermore, as one moves toward the backbone of the network traffic can be aggregated into intermediate flows. It is fairly easy to show that combining PDUs means that the fraction of the PDU consumed by PCI remains constant or decreases. Also remember that even when the architecture has more layers, the number of layers seen by any router is never more than 2 or 3. IOW, the flows in any one layer don't see any more overhead than they do now. The host might see more in the case of mobility. However, again because each layer uses the same protocol with different policies, it can be implemented by one state machine with different policy vectors for each layer. PDUs are not passed between layers.[14] The state machine processing the PDU treats the PDU as a tape as in a Turing Machine, stepping through the PDU processing protocol layers as needed. It is considerably more efficient than current implementation strategies.

3) *Not lose data.* Why would it lose data? The same flows are in place through the lifetime of the application connection. There are no tunnels to set up and tear down. Data would only be lost if there was no physical communication with the network.

4) *Manageability.* The application names never change. The mapping of application name to (N)-address and of (N)-addresses to (N-1)-addresses is ensured by engineering the scope of the layers to ensure the update time is small compared to the rate of change of (N-1)-addresses. Also, in this model address

[12] The traditional definition of mesh network is any network where the graph has arbitrary connectivity whether wireline or wireless.

[13] We have not found a situation that would require more than 32 or 48 bit addresses unless one simply wanted to represent greater granularity in the hierarchy and was unconcerned with the amount of unused address space.

[14] In an architecture of 'hand-crafted" protocols, passing PDUs between layers and fixed layers is almost required. Common protocols avoids this constraint.

resolution does not work as it does in the Internet. In the Internet, the client does a DNS request and if the domain name is found, it returns an address to use in the initiating TCP connection. But following the IPC Model, the resolution of the name to address mapping confirms that the requested application is there and that the requestor has access to it before returning a response.[15]

5) *Scales.* This definitely scales and scales indefinitely. The only constraints are imposed by physics. In fact, this is probably the only approach that does scale to the densities that are expected.

5. Conclusions

Clearly mobility is a much simpler problem to solve when one uses a complete architecture rather than one that is missing more than half of the minimum required elements. The other aspect that must be recognized is that the capability does not require any additions to the architecture. Mobility in RINA doesn't require setting up tunnels, re-writing packet headers or using special protocols: it is just achieved by utilizing the tools the architecture provides and that are used for normal operation, albeit using them a bit more frequently, a combination of routing updates, changing addresses of IPCPs and designing the number and size of layers in different parts of the network to accommodate the load, scale, and rate of change of the (in this case) mobile terminals to be supported. But again, nothing more than what one would do to design a network for any other purpose. All standard procedures that can be performed in any RINA network, mobility is no special case.

The fact that many significant capabilities are merely a consequence of the architecture is somethign we find over and over. In this case, it doesn't just provide a *solution* for mobility, but also *solves* multihoming, reduces router table size, simplifies multicast, and is more secure and with less overhead than any of the proposed Internet solutions. It also has additional benefits. It greatly simplifies processing, eliminates the need for complex rules for firewalls, eliminates the need for expensive Deep Packet Inspection, and there aren't a multitude of address forms. This will also yield a more well-behaved network. All of this means that much more attention can be given to using the network, dare we say, profiting from the network and much less time and effort. just keeping it up.

6. References

1. **Egli, Peter.** http://www.slideshare.net/PeterREgli/p-6098167). 2011
2. http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mob_pmipv6/configuration/xe-3s/deployment/mob-pmipv6-deploy/Proxy_Mobile_IPv6_Network-Based_Mobility.html#GUID-B4625DCE-8ACC-4171-9A65-D11A59675E95
3. J.Lee, J. Bonnin, P. Seite, and H. A. Chan, "Distributed IP mobility management from the perspective of the IETF: motivations, requirements, approaches,

[15] Watson's results also imply decoupling port allocation and synchronization. This also implies this approach and also improves security and thwarts many spoofing attacks.

comparison, and challenges," IEEE Wireless Communications, vol. 20, no. 5, pp. 159–168, October 2013.

4. F. Giust, L. Cominardi, and C. Bernardos, "Distributed mobility management for future 5G networks: overview and analysis of existing approaches," IEEE Communications Magazine, vol. 53, no. 1, pp. 142–149, January 2015.