PeerNet: Pushing Peer-to-Peer Down the Stack

Jakob Eriksson, Michalis Faloutsos, Srikanth Krishnamurthy [*]
University of California, Riverside

Abstract - An unwritten principle of the Internet Protocol is that the IP address of a node also serves as its identifier. We observe that many scalability problems result from this principle, especially when we consider mobile networks. In this work, we examine how we would design a network with a separation between address and identity. We develop PeerNet, a peer-to-peer-based network layer for large networks. *PeerNet is not an overlay on top of IP*, it is an alternative to the IP layer. In PeerNet, the address reflects the node's current location in the network. This simplifies routing significantly but creates two new challenges: the need for consistent address allocation and an efficient node lookup service. We develop fully distributed solutions to address these and other issues using a per-node state of $O(\log N)$, where N is the number of nodes in the network.

PeerNet is a radically different alternative to current network layers, and our initial design suggests that the PeerNet approach is promising and worth further examination.

1 Introduction

How would we design a network layer with mobile nodes and peer-to-peer interactions in mind? This question would have seemed like a theoretical exercise a few years back, but it has become legitimate, if not necessary, with the current technological trends and commercial initiatives. In fact, we observe an overwhelming popular and commercial interest in mobile wireless connectivity [1, 7], consumer owned networks [3, 2, 4] and mesh networking [5, 6]. A vision that we share with [12] involves pockets of peer-to-peer wireless connectivity interconnected with traditional wired lines. Implementing such a vision introduces new networking requirements, and we may need a novel network architecture. Here, we focus on the network layer of such a future architecture.

Although the Internet Protocol (IP) has been a spectacular success, this should not prevent us from assessing its suitability for the networks of the future. Most of the above initiatives seem to rely on the Internet Protocol. An implicit principle of this protocol is that the IP address of a node is tightly coupled with its identity. This has worked well so far, since the Internet supports mostly stationary nodes with wireline links and well defined consumer-provider relationships. However, the *address as identifier* paradigm may encounter problems in highly mobile networks. We have already seen mobility and scalability push IP to its limits. While it is possible to satisfy most new requirements with patches such as NAT, DHCP Mobile IP, the end result is seldom elegant and often plagued by new problems. One striking example of the above paradigm failing is the fact that a TCP session will break if one of the end points changes its address. This is an important issue as networks are becoming increasingly mobile.

The overarching question is how we would design the network layer for future networks if we were to start from scratch. Our target environment is very large potentially wireless and mobile networks. Therefore, we want an agile, plug and play, fault-tolerant, and scalable networking layer. A source of inspiration is application layer peer-to-peer networking, an area that has seen tremendous advancements recently. Therefore, we pose the question: what can we gain by bringing the peer-to-peer concept from the application layer down to the networking layer? We develop a set of guidelines that seems to satisfy our networking vision. We want the new network layer to: *a) minimize the need for manual configuration, b) avoid centralized solutions and node specialization in favor of distributed and peer-to-peer solutions, and c) localize control overhead.*

In this paper, we present PeerNet, a network layer with integrated support for routing between peers. PeerNet makes an explicit distinction between node identity and address. The address of a node reflects

[*]This material is based upon work supported by the National Science Foundation under CAREER Grant No. 9985195, DARPA award FTN F30602-01-2-0535.

its current location in the network at all times. This simplifies routing but introduces two new challenges. First, we need a node lookup service that will provide the address for a given a node identifier. Second, PeerNet needs to maintain addresses dynamically: as a node moves, its address changes to reflect the new location. Our initial design suggests that PeerNet is feasible and potentially fundamental component of our vision for future networks.

Our work in perspective. PeerNet is a radically new architecture that brings peer-to-peer concepts to the network layer. Although the design presented here is not complete, it provides the backbone and several non-trivial algorithmic solutions. In our upcoming implementation, we expect to complete and finetune our design. Furthermore, we expect this work to provoke a constructive reevaluation of current networking architectures.

Related work. Area Routing [11] and Landmark routing [16] are classical papers on hierarchical routing. LANMAR [13] and L+ [9] are modern extensions of Landmark routing, whereas PeerNet to our knowledge is the first protocol for dynamic networks with similarity to Area Routing. For a survey of ad hoc routing, see [10], and for a survey of distributed hash tables, see [14]. Recently, several efforts such as [15, 8] address some of the same issues as PeerNet but do so by adding functionality to the existing IP infrastructure.

2 Operations Overview

The key idea in PeerNet is the separation of the *identity* and the *address* of a node. For now, we can assume that each PeerNet node has one unique identifier and one unique address. The address is dynamic and changes with node movement to reflect the node's location in the network. The ID of the node remains the same throughout, reliably identifying the node despite address changes. An integrated distributed node lookup service maps identifiers to addresses.

Joining the network. To join a PeerNet network, a node establishes a physical connection to at least one node already in the network and requests an address. The receiving node(s) answer(s) with an available address. The joining node then "registers" its identifier together with the address in the distributed node lookup service. As a node moves, it requests and receives new addresses from its new neighbors. On each address change, the node updates its entry in the lookup service.

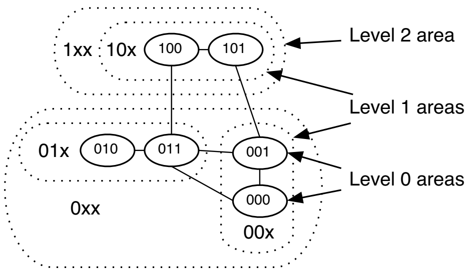


Figure 1: **A 3-level area network. Level-0 areas are single nodes and Level-n areas consist of up to two Level-$(n-1)$ areas.**

Packet routing. The sender node only needs to know the *identifier* of the receiver. Before sending its first packet to some destination, the sender looks up the current address of the destination node using the lookup service. PeerNet packets contain both the identifier of the destination and the last known address and routing is done in a Distance Vector fashion[1], one hop at a time. If the destination cannot be reached, the lookup table is consulted along the way to find the new address of the destination.

3 The PeerNet Network Layer

PeerNet targets networks consisting of a large number of mobile and stationary nodes, connected by bidirectional links using any current MAC technology.

Node addresses are dynamically assigned depending on the node's current position in the network. More specifically, the addresses are organized as leaves of a binary tree. We call this the *address tree*. By selecting node addresses carefully, we guarantee that nodes within a subtree are able to communicate using only nodes inside that subtree. We will also use the term *area*, which we define as follows:

Area: An area is a set of network nodes such that for every pair of nodes in the area, there exists a path between them that consists only of nodes in the area.

A subtree of the address tree is an area, and we will use these terms interchangeably[2].

[1]Note this is not regular Distance Vector routing. Routing entries point to predetermined positions in a virtual hierarchy of nodes, thereby reducing the size of the routing table to $O(logN)$.

[2]Not all areas correspond to an address subtree, but we are not

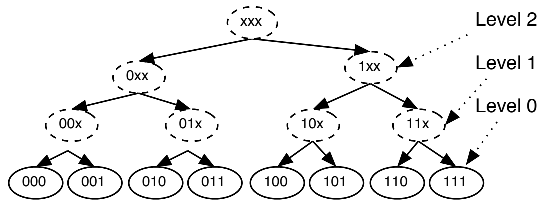


Figure 2: **3-bit address space as a binary tree. Physical nodes exist only at the leaf level.**

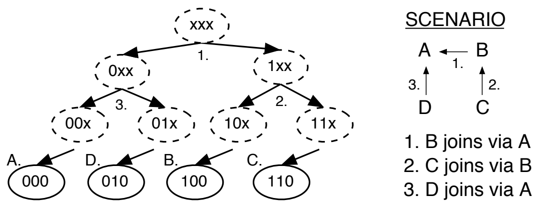


Figure 3: **Address tree for a small network topology. The numbers 1-3 show the order in which nodes were added to the network.**

As shown in figure 1, a Level-0 area is a single network node. A Level-n area is recursively defined as consisting of two connected Level-$(n-1)$ areas. The following is crucial to all PeerNet operations:

PeerNet Area Invariant: All nodes belong to a nested sequence of areas, one area of each level. All nodes within an area share a unique address prefix.

In figure 1, each node belongs to a Level-0, 1 and 2 area respectively, with the Level-0 area being the node itself. In a PeerNet with addresses consisting of l bits, there would be l corresponding area levels. To maintain the area invariant, a node that violates the invariant resigns from the network and request a new address from a neighboring node.

The PeerNet network layer consists of three major parts. First, *address allocation* maintains one address per node, in compliance with the area invariant. Second, *routing* disseminates enough information about the global state of the network and uses this information to efficiently deliver packets to their destination. Third, *node lookup* is a distributed network service mapping a node identifier to its current network address. We describe these areas in more detail below.

Finally, there are several issues that the current paper has not attempted to address. Security is one such issue which we are currently examining, other issues include the performance effects of network partitioning and merging.

3.1 Address Allocation

PeerNet assigns addresses to nodes dynamically, so that the PeerNet area invariant is preserved. For increased efficiency and stability, address allocation must result in a well balanced address tree and nodes within an area should be well connected.

A PeerNet address consists of l bits. These bits describe a position in a corresponding l level binary address tree. Figure 2 illustrates the idea for $l = 3$. All physical nodes reside at the leaf level, and hold complete l-bit addresses.

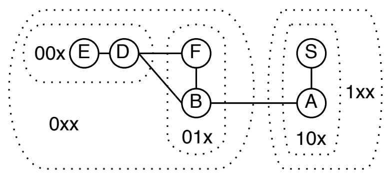
To join a network, a new node requests an available address from an existing PeerNet node. Finding an available address can be implemented in different ways, attempting to balance two somewhat conflicting factors. On the one hand, we want the address allocation to be based on local information. This way, we can improve the scalability of the network and minimize the necessary control overhead. On the other hand, we would like to utilize the address space efficiently. We present our framework to strike the balance in this trade-off. Our solution hides several subtleties; we only provide a high level overview here.

Obtaining an address. We develop a protocol hat enables nodes to allocate addresses in a local and conflict free fashion. At any given time, each node ”manages” a range of addresses including its own address. Every time a new node requests an address, the responding node splits its *address range* in half, and delegates control of the upper half to the new node. The address of the new node is set to be the lowest address within the delegated range. In this manner, nodes are evenly distributed throughout the address space.

Figure 3 illustrates this procedure for 3-bit addresses. Node A starts out alone and has address 000. Furthermore, it controls the entire 3-bit address space. When node B joins the network, it is assigned the address 100. At this time Node A can no longer assign addresses that begin with ’1’. Similarly when C joins the network by connecting to B, it gets assigned address 110 and Node B is then precluded from assigning the address 111. Finally, when D joins via A, it gets assigned address 010.

The characteristics of the address allocation can have a major impact on the performance of the network. For example, if an available address can not be

interested in such areas, and will not refer to them in this paper.



ADDRESSES AND ROUTING TABLES

```
Node D = 001    Node S = 101
 Level 2 : B     Level 2 : A
 Level 1 : F     Level 1 : -
 Level 0 : E     Level 0 : A

Node B = 010    Node A = 100
 Level 2 : A     Level 2 : B
 Level 1 : D     Level 1 : -
 Level 0 : F     Level 0 : S
```

Figure 4: **In a 3-level network, each node keeps 3 routing entries. Those of S, A, B and D are shown on the right.**

found, the request is refused. However, with a large enough address space and with efficient address tree maintanance, this is unlikely to happen. Two issues are critical for the address tree: a) we want to keep the address tree balanced, and b) we want to maximize the connectivity within an area. These two objectives may at times be conflicting, and we are currently evaluating techniques to find a good balance between tree balancing and inter-area connectivity.

Address tree balancing. We need a way to balance the tree while maintaining the PeerNet area invariant. If a particular area becomes congested, using up all locally available address space, new nodes that try to obtain an address may be unable to do so. Thus, in order to alleviate cases of local congestion in the tree, we would like nodes to proactively migrate in the tree in order to balance it. Migrating in this case, means simply to select a new address; without affecting connectivity and within the constraints of the area invariant.

Maximizing the intra-area connectivity. We want to select addresses in such a way that nodes within an area are well connected by physical links. This improves the routing performance and tolerance to link failures, and is especially desirable in mobile networks.

3.2 Routing

Intuitively, PeerNet routing can be seen as recursive procedure descending through the address tree. At each level, we decide through which of the two available subtrees to descend[3], starting from the top (the most significant bit). At the physical level, a packet is routed one level at a time. When the packet has reached any node in the correct subtree on one level, the packet is routed to the nested lower level subtree, that contains the destination node. Given the PeerNet area invariant, every step takes us closer to the destination in both the network topology and along the address tree. Using a distance vector approach, the PeerNet routing table of each node requires only $l = \log N$ entries, that is, one entry per level in the address tree.

Let us revisit the address tree (figure 2) in order to understand some subtleties of PeerNet routing. For addresses of $l = 3$ bits, the entire address space can be represented by xxx, where $x \in \{0, 1\}$. The most significant bit divides the address space in two subtrees, $0xx$ and $1xx$. We refer to these as Level-2 subtrees. Similarly, the second bit divides the left Level-2 subtree, into the two Level-1 subtrees 00x and 01x. We refer to such pairs of trees as *siblings*. Furthermore, we define a k-**sibling** of a node to mean the sibling subtree of the Level-k subtree to which the node belongs. In figure 2, 1xx is the Level-2 sibling of 010, whereas 00x is the Level-1 sibling of 010.

To route a packet, a PeerNet node will compare the destination address with its own address. If the most significant bit differs in the two addresses, then the destination is in the "other half" of the address space, the $(l - 1)$-level sibling subtree. In general, a node compares its own address and that of the destination one bit at a time, starting with the most significant bit. If it finds a bit that differs, say the i^{th} bit, it will forward the packet towards the i-sibling. Eventually, all the bits will be identical, at which point the destination is reached.

The routing table. Every node maintains a routing table that has l entries. The k^{th} entry in this table contains the next hop to the k-sibling subtree.

As an example, let us assume that node S[101], where 101 is the address of the node S, wants to route a message to D[001] in figure 4. Given the difference in the most significant (Level-2) bit , S[101] forwards the packet to A[100], as indicated by S[101]'s Level-2 routing table entry. Similarly, A[100] looks at the

[3]Although this is an intuitively appealing statement, we have to keep in mind that the address tree is not an actual network. In fact, only the leafs of the tree correspond to actual nodes. The internal nodes of the tree do not correspond to actual nodes.

most significant (Level-2) bit and forwards the message to node B[010], which is inside the Level-2 area 0xx. B[010] and the packet destination D[001] have the same Level-2 bit which means that the message is in the correct Level-2 area. Node B[010] looks at the next (Level-1) bit of the address, and forwards the message according to the Level-1 routing entry, which points to D[001]. Once there, D[001] looks at the third (Level-0) bit and realizes that it is the recipient.

PeerNet uses a distance vector approach in updating its routing tables. Although link state routing is also an option, we choose distance vector for its low overhead, low computational cost per router, and ease of implementation. PeerNet can be made cycle free with little overhead, thanks to the area invariant. It can be shown that cycle-free PeerNet routing requires $O(l) = O(logN)$ per-node state, but we will omit the details due to space constraints.

3.3 Node Lookup

Since the ID of a node is not its address, PeerNet provides a distributed node lookup service for looking up an address given an identifier. Intuitively, each identifier is mapped though some function to a single address and the node that currently controls that address is required to store the mapping. Let us initially assume that an identifier is an m-bit number $m \geq l$ that uniquely identifies a node. In contrast, an address is an l-bit number that determines a position in the network topology. We call the pair $(identifier, address)$ a *lookup entry*.

Mapping entries to nodes. In the basic case, each lookup entry is globally mapped onto a single node. This node is responsible for storing the entry and responding to requests for that entry. In our current design, the node is chosen so that its address minimizes the integer value of the expression $nodeAddress\ XOR\ identifier$[4]. We call this the **xor-distance criterion**. Finding the minimum *xor-distance* node for an identifier is a process similar to packet routing, where an update is routed towards the identifier rather than a particular address. We observe that the cost of performing this mapping is the same as that of regular packet routing, but omit further details due to space constraints.

[4]Since $m \geq l$ the expression is applied only to the l most significant bits of m.

Challenges due to node movement. Moving nodes present two challenges to the lookup service. First, node movement leads to address change which means lookup entries have to be updated. Second, when the address of a node changes, it is likely that some of its lookup entries need to be moved to a new node according to the *xor-distance criterion*. The area invariant guarantees that the new node will be in the vicinity of the moving node, so the second issue is a minor one.

If we were to simply use *xor-distance* mapping without modification, any locality in the communication patterns would be lost. An example of this is node X looking up the address of node Y, which happens to be just a few hops away. Y's lookup entry could potentially be mapped to a very distant node, forcing X to perform a long-distance lookup to set up short-distance communication with Y. Similarly, nodes that change address would occasionally have to send their updates long distances, which may be too expensive in very mobile scenarios. We will now briefly describe how to address these problems.

Solution A: Preserving locality of lookups. Instead of storing the entry in a single location in the network, we store it in multiple locations. These locations are chosen so that a lookup initiated in the vicinity of the desired node will find a locally stored entry instead of sending a query across the whole network. More specifically, the extra entries are stored in nodes that satisfy *minimum xor-distance* with 1 or more of the most significant bits removed from both address and identifier. Our improved locality preserving lookup starts with a very local scope and iteratively tries larger and larger subtrees until the entry is located.

Solution B: Creating locality of updates. We observe that a moving node is likely to change its lower order bits more frequently than the higher order bits. We can use this to reduce the cost of updating the entries of moving nodes. Instead of storing the whole address in the abovementioned locations, local nodes keep the less significant bits of the address, while more remote nodes store the more significant bits of the target address. This way, non-local nodes will need less frequent updates.

The lookup process is further modified so that once the more significant bits of the address have been located, the query is forwarded to that area. Once in the local area, additional less significant parts of the address can be located. We are currently studying the tradeoff between additional lookup steps and increased locality of communication.

Identifiers as communication abstractions. By allowing identifiers to map to more than one address, we can efficiently implement **multicast** and **anycast** in PeerNet. In doing that, we extend the role of the identifier from that of uniquely identifying a node, to that of uniquely identifying a node *or a group of nodes*. A node would subscribe to a multicast group by adding a mapping between the multicast group identifier and its current address. Similarly, anycast can be implemented by mapping a single service identifier to a set of service providing nodes. A local lookup of the anycast identifier would then return the closest node providing the requested service.

3.4 Implementation and Deployment

In our ongoing implementation, we have decided to use addresses of size $l = 128$, which is the size of the IPv6 address. As a result, the corresponding routing table size is 128 entries. For maximum portability, we are developing the kernel code in C, and expect to release kernel modules for Darwin/MacOS X and Linux initially.

Leveraging the existing infrastructure. PeerNet does not need the support of a wireline infrastructure, but we want to be able to leverage any available infrastructure. We envision tunneling through the Internet to interconnect disconnected PeerNet networks initially and when available. In addition, we want to develop protocols and tools to facilitate the communication between PeerNet and Internet nodes. We also want to enable TCP/IP emulation for IP-only software, which we may want to run on PeerNet nodes.

4 Conclusion

We present PeerNet, a radically different network layer. Recent trends in wireless technology, popular demand, and commercial interest impose new requirements on networks. This has prompted us to reevaluate the role of IP in future networks. We envision dynamic networks with pockets of peer-to-peer wireless connectivity interconnected with traditional wired lines.

There are two fundamental and complementary novelties in PeerNet. First, there is a distinction between the identity and address of a node. This distinction enables us to handle mobility in a novel way, improving the scalability of the system. Specifically, the effect of node mobility is confined to the neighborhood of a moving node in most cases. Second, PeerNet supports peer-to-peer routing at the network layer. Critical functions like address allocation and routing are addressed in a distributed and cooperative fashion using a per node state of $O(\log N)$. In addition, PeerNet requires no manual configuration, and is fully distributed.

To summarize, PeerNet is a promising new network layer with the potential to scale to very large dynamic networks. In our ongoing implementation, we will validate and refine the ideas presented here.

References

[1] Boingo inc. www.boingo.com.

[2] Community wireless. www.communitywireless.org.

[3] Consume.net project: Trip the loop, make your switch, consume the net! www.consume.net.

[4] Freenetworks. www.freenet.org.

[5] Mesh networks inc. www.meshnetworks.com.

[6] Mitre mobile mesh. www.mitre.org/tech_transfer/mobilemesh/.

[7] Starbucks inc. www.starbucks.com.

[8] M. Beck, T. Moore, and J. S. Plank. An end-toend approach to globally scalable network storage. In *SIGCOMM*, August 2002.

[9] B. Chen and R. Morris. L+: Scalable landmark routing and address lookup for multi-hop wireless networks, 2002.

[10] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE NETWORK*, 16(4), 2002.

[11] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks: Performance evaluation and optimization,. *Computer Networks*, 1, 1977.

[12] N. Negroponte. Being wireless, 2002. www.wired.com/wired/archive/10.10/wireless.html.

[13] G. Pei, M. Gerla, and X. Hong. Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. In *ACM MobiHOC'00*, 2000.

[14] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. IPTPS, 2002.

[15] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *ACM SIGCOMM 2002*, August 2002.

[16] P. F. Tsuchiya. The landmark hierarchy : A new hierarchy for routing in very large networks. In *SIGCOMM*. ACM, 1988.