TurfNet: An Architecture for Dynamically Composable Networks

Stefan Schmid, Lars Eggert, Marcus Brunner and Jürgen Quittek

NEC Europe Ltd.
Network Laboratories
Kurfürstenanlage 36
69115 Heidelberg, Germany
{schmid, eggert, brunner, quittek}@netlab.nec.de

Abstract. The Internet architecture is based on design principles such as end-to-end addressing and global routeability. It suits relatively static, well-managed and flat network hierarchies. Recent years have shown, however, that the Internet is evolving beyond what the current architecture can support. The Internet architecture struggles to support increasingly conflicting requirements from groups with competing interests, such as network, content and application service providers, or end-users of fixed, mobile and ad hoc access networks. This paper describes a new internetworking architecture, called TurfNet. It provides autonomy for individual network domains, or Turfs, through a novel inter-domain communication mechanism that does not require global network addressing or a common network protocol. By minimizing inter-domain dependencies, TurfNet provides a high degree of independence, which in turn facilitates autonomic communications. Allowing network domains to fully operate in isolation maximizes the scope of autonomic management functions. To accomplish this, TurfNet integrates the emerging concept of dynamic network composition with other recent architectural concepts such as decoupling locators from identifiers and establishing end-to-end communication across heterogeneous domains.

1 Introduction

The Internet has evolved from a small research network to a huge, worldwide information exchange that plays a central role in today's societies. A growing diversity of interests in this global internetwork (*e.g.*, commercial, social, ethnic, governmental, etc.) leads to increasingly conflicting requirements among competing stakeholders. These conflicts create tensions the original Internet architecture struggles to withstand.

As one example of an ongoing "tussle" [1], consider the commercial success of the Internet. It has created a large number of competing service providers that aim to outperform one another in order to increase their profits. The result is an increased willingness to forgo agreed-upon standards that allowed a more cooperatively managed Internet to succeed. This paper argues that despite the remarkable success of the

Internet architecture – often attributed to its robust design principles – its underlying assumptions no longer fully match today's networking requirements. In particular, specialized new types of networks, such as sensor networks, mobile *ad hoc* networks and the widespread deployment of "middleboxes" have begun to stretch the capabilities of the existing architecture. This has prompted research into fundamentally different network architectures, such as FARA [2], Plutarch [3], Triad [4] or IPNL [5].

This paper proposes a new internetworking architecture called *TurfNet*, which addresses the limitations of the Internet architecture by accommodating conflicts of interests among different stakeholders and supporting their diverse interests.

The *TurfNet* architecture focuses on interoperation between otherwise autonomous networks. These autonomous networks are modularized according to the inherent boundaries drawn by the different interests of the stakeholder involved. This paper uses the name *turf* to denote such an autonomous network. The term *turf* has an innate connotation to ownership and responsibility that the *TurfNet* architecture reflects. Other papers introduce different terms for similar concepts, such as *regions* [6] or *contexts* [3]. The concept is also related to the Internet's *Autonomous Systems* (AS).

Isolated, autonomous *TurfNets* dynamically compose into new, larger autonomous *TurfNets* that integrate the original networks. The process of dynamic network composition supports the interconnection of heterogeneous networks, such as mobile and *ad hoc* networks, IPv4 networks or IPv6 networks. Composed "super" networks manage this integration by abstracting potential isolation (*e.g.*, over-lapping address spaces) or heterogeneity (*e.g.*, incompatible network protocols) issues among the constituent subnetworks. One mechanism for supporting this heterogeneity is address and protocol translation, but the architecture supports other, equivalent mechanisms as well.

Backwards compatibility with today's Internet is a crucial requirement for any next-generation internetworking architecture. This was arguably one critical mistake during the design of IPv6. The *TurfNet* architecture maintains compatibility with the current Internet architecture by supporting it as one specific network type, along with 3G mobile networks, *ad hoc* networks or sensor nets.

The first part of this paper motivates this research and discusses the underlying design principles of the architecture. Section 3 then outlines the *TurfNet* architecture and explains how it addresses the "new" needs of today's networking requirements. Section 4 describes the basic end-to-end communication across several layers of composed *TurfNets*. Section 5 then discusses the scalability properties of the *TurfNet* architecture. Finally, the remaining sections compare and contrast the *TurfNet* architecture against other related work and conclude with an outlook on future work.

2 Design Axioms

This section briefly discusses the basic axioms of the *TurfNet* architecture.

Packet switching: Packet-switched networks increase performance and efficiency by multiplexing bursty traffic from different sources onto the same medium. Furthermore, packet switching provides a simple, generic communication framework that supports many different kinds of data flows and requires little explicitly managed state inside the network.

Separation of identity and location: Today's IP addresses denote both the identity of a node as well as its topological location. Several proposals for splitting the two functionalities exist, and the *TurfNet* architecture will adopt this important new concept with a focus on supporting mobility.

Global namespace: The Internet's *Domain Name System* (DNS) [7] is a global, hierarchical namespace based on *Fully Qualified Domain Names* (FQDNs). Other current naming schemes, such as the ones used for the *Host Identity Protocol* (HIP) [8] or the *Layered Naming Architecture* [9], also make use of globally unique names or identities.

Similar to these approaches, the *TurfNet* architecture globally identifies network entities belonging to different, autonomous *TurfNets.* Consequently, the *TurfNet* architecture could either use FQDNs, HIP identities, or any other global namespace – or even different global namespaces at the same time. For simplicity, the remainder of this paper uses the generic term "name" to refer to arbitrary types of global identifiers.

Flexibility in business models: Networking moves from a few monolithic operators to a scenario where the competing interests of owners, roaming brokers, transit network operators, users, and service providers, among others, must be accommodated and balanced. Consequently, a future internetworking architecture must enable, support and manage new business models and complex value chains.

Autonomous *Turfs*: Braden [10] proposes the meta-architectural principle that different regions of the network should be allowed to differ from each other: "minimize the degree of required global architectural consistency." This paper adopts this principle as a necessary enabler for future businesses and diversity between domains.

Inter-*Turf* control interface: Network control must cross domain boundaries, for example, to support address registration and name lookups across individual *TurfNets*. Such functions require a common, high-level inter-*Turf* control interface to exchange control state and configuration information. This facility must not be tied to a specific network protocol – which could be different within individual *TurfNets* – but rather depend on a common data format.

3 The *TurfNet* Architecture

A *TurfNet* is a completely autonomous network domain. To achieve autonomy, every *TurfNet* encompasses its own, independent network addressing mechanism and all associated control plane functions, such as routing protocols, name-to-address resolution, etc. A common, shared namespace to enable inter-*Turf* communication is the only global requirement (apart from the high-level inter-*Turf* control interface). In contrast to today's Internet architecture, *TurfNets* do not rely on globally shared state or pervasive functionality like a common network protocol, a globally shared address space or a global name service.

Another fundamental design choice that supports autonomy of *TurfNets* is the concept of encapsulation. It allows *TurfNets* to fully hide their internal characteristics, structures and policies. Such a modular network architecture allows individual players with potentially competing interests to interoperate in a controlled and protected manner and thus better suites the new requirements of future network communication.

If a *TurfNet* chooses to hide its internals (*e.g.*, network addresses and protocols), external nodes cannot directly communicate with individual nodes of that *TurfNet* anymore. Communication without knowledge of the peer node's local network address (and protocol) requires new network capabilities. In the *TurfNet* architecture, nodes first have to acquire a *Turf*-local representation in the destination *TurfNet.* In essence, each *TurfNet* maps the remote communication peers into part of its local address space. To other local *TurfNodes*, remote nodes appear to be of the local *Turf.*

3.1 Architecture Overview

Figure 1 shows an abstract view of the proposed *TurfNet* architecture. Its key components are:

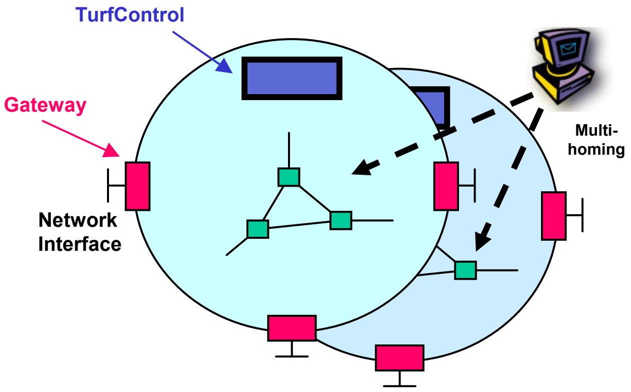

Figure 1. The *TurfNet* Architecture.

***TurfControl*:** The *TurfControl* is a logical, per-*Turf* entity comprised of a *TurfNet's* essential control functions and services. It encompasses all traditional control plane functionalities in the network, such as address allocation, routing and name resolution. It further includes the new *TurfNet* functionality, for example, to manage composition.

A *TurfNet* handles all its control functionality locally. This is an important prerequisite for maintaining the autonomy of individual *TurfNets*. Because of the importance of the *TurfControl* for the proper operation of a *TurfNet*, it must be resilient. In the case of large (composed) *TurfNets*, distribution and replication of this logical functionality across many nodes will improve scalability as well as resilience.

***TurfNode*:** A *TurfNode* is a network node in a specific *TurfNet*. It communicates with *Turf*-local network protocols and uses local addressing and routing mechanisms.

A *TurfNode* interacts with the local *TurfControl* for all control plane operations, such as address allocation, routing or name resolution.

To support multi-homing as a fundamental part of the *TurfNet* architecture, physical nodes may concurrently participate as full-fledged logical *TurfNodes* in multiple *TurfNets* (see Figure 1). Note that multi-homed nodes do not necessarily act as gateways between the different *TurfNets* (see below).

Gateways: *TurfNet* gateways are special, multi-homed nodes. Besides being part of multiple *Turfs* at the same time, they also actively relay traffic between the different *TurfNets* they are a part of. To enable this functionality, such gateways must be fully operational *TurfNodes* in both *TurfNets* (*i.e.,* they must be able to communicate and have at least one interface in both *TurfNets*).

The main responsibility of these gateway nodes is to relay traffic between the different *TurfNets*. In the case of peering, *TurfNets* use independent network addressing or even different network protocols. Gateways will then also perform the required address and protocol translations. For example, a gateway between IPv4 and IPv6 *TurfNets* will translate between the two network protocols and their respective address spaces. If two *TurfNets* use the same protocols and have compatible addressing, the gateway will simply forward data packets – acting like a traditional Internet router. Whether a *Turf* gateway acts as a traditional router, as network address translator, or even as protocol translator depends on its local environment. Section 4 discusses advantages and disadvantages of the different roles further.

Another task of gateways is to connect the *TurfControls* of the peering *TurfNets*. If *TurfNets* use different control protocols, the gateway must translate control messages.

3.2 Network Composition

The *TurfNet* architecture adopts the central architectural principle of network composition from the Ambient Networks project [11]. Network composition is a new paradigm that allows cooperative networks to automatically negotiate inter-working agreements to establish inter-domain communication. A related paper discusses network composition at the control plane level to facilitate self-organization [12]. This paper considers network composition as a means to interconnect fully autonomous networks, *i.e.*, *TurfNets.* It focuses on inter-domain communication among heterogeneous networks, including different network protocols and address spaces.

TurfNets can dynamically compose with each other to form new, integrated or interconnected *TurfNets*. Two different variants of this operation are possible, resulting in *horizontal* or *vertical* composition of individual networks.

3.2.1 Horizontal Composition

When multiple *TurfNets* merge into a single *TurfNet* such that they share a common control plane as a result, they compose *horizontally*. This type of composition fully integrates the original *TurfNets* into the final, merged *TurfNet*. For example, one *TurfNet* could adopt the addressing mechanisms and protocols of the others, or the merging *TurfNets* could all agree on a new set of addressing mechanisms and protocols. Figure 2 illustrates this process.

A key characteristic of horizontal composition is that merged *TurfNets* have a single logical *TurfControl* instance. The original *TurfNets* lose their "identity" after the composition and appear from then on only as part of the identity of the new *TurfNet*. The process of horizontal composition is irreversible – no information about the original constituents exists that would allow decomposition into the original *TurfNets*.

Despite this loss of identity, horizontally composed networks can still split. This occurs, for example, when a *TurfNet* becomes partitioned due to network link failures. However, partitioning will not typically restore the original *TurfNets* but instead result in an arbitrary set of *TurfNets*.

Finally, note that the *TurfNet* architecture does not prohibit a *TurfNet* from being structured into different administrative domains. These domains, however, are management entities that are not visible in an architectural description of a *TurfNet*.

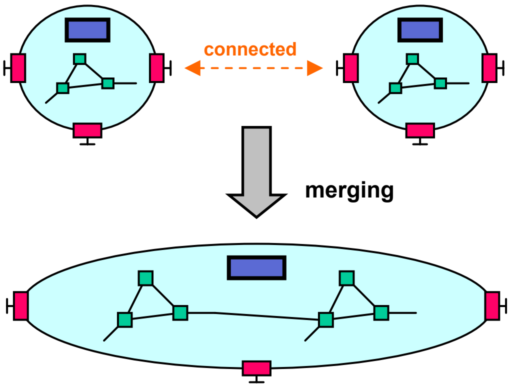

Figure 2. Horizontal composition of *TurfNets*.

3.2.2 Vertical Composition

Vertical composition, on the other hand, is the process by which *TurfNets* compose such that each individual *TurfNet* preserves its autonomy (with respect to addressing, routing, name resolution, etc.) even after it becomes part of the newly composed *TurfNet*. In this case, two *TurfNets* are said to compose vertically such that one becomes the governing parent *TurfNet* of the other. Figure 3 illustrates this case and Figure 4 highlights hierarchical structure of this composition variant.

The advantage of vertical network composition is encapsulation of administrative, control and routing functionalities, as well as isolation of internal structures. Because of hierarchical composition, new sub-*TurfNets* may join locally, without requiring global interaction. This reduces the complexity of administrative and control negotiations.

Similar to the proposed *TurfNet* architecture, the Internet also contains administrative domains that hide the internal complexity of a domain (*e.g.*, intra-domain routing). However, the main difference to the *TurfNet* architecture is that its vertical composition is not limited to two levels. Furthermore, vertical composition fully separates administrative and control functionalities (*i.e.,* not even the addressing or routing must be globally agreed upon).

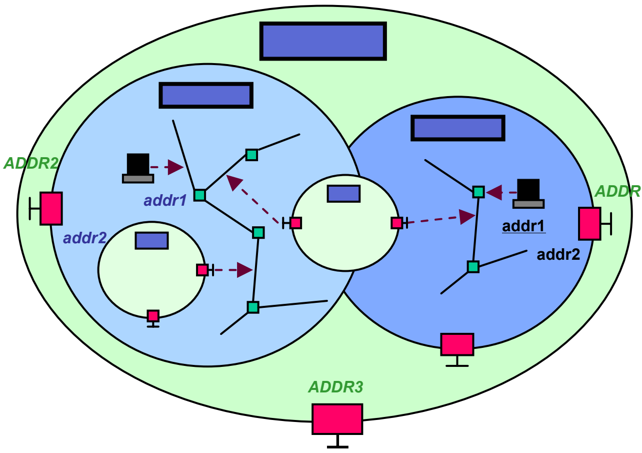

Figure 3. Vertical composition of *TurfNets*.

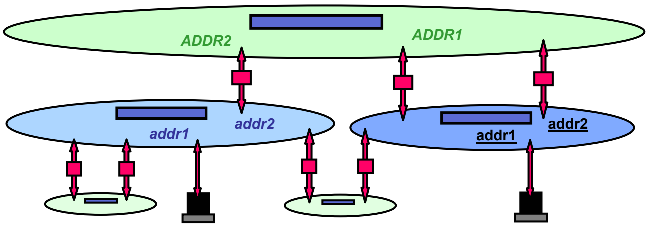

Figure 4. Hierarchically structured *TurfNets* after vertical composition.

Another important advantage of vertical composition is that the transition from today's networking infrastructure requires few changes to existing networks. Specialized gateway nodes facilitate interoperability and integration by translating between different domains. For example, vertical composition can integrate existing IPv4 and IPv6 networks without modifications to existing protocols and protocol stacks.

Communication across vertically composed *TurfNet* boundaries occurs through well-defined gateways, which relay traffic between the different *TurfNets*. If the peering *TurfNets* use the same network protocol and non-overlapping network addresses, a gateway simply forwards packets between the domains, similar to traditional Internet routers. However, if *TurfNets* use different network addressing schemes or different network protocols, a gateway also performs bidirectional network address and protocol translation.

The number of gateways used to compose a *TurfNet* into a parent *TurfNet* depends on reliability and scalability requirements. Large volumes of traffic require a larger number of gateway nodes, to share the processing load of address translation. Multiple gateways also provide resilience in the case of gateway failures.

As illustrated in Figure 4, *TurfNets* can simultaneously compose with several higher-layer *TurfNets.* This is especially important for multi-homed *TurfNets* that peer with several providers.

3.3 Discussion

The previous, brief description of composition approaches has illustrated that horizontal and vertical composition are fundamentally different and address different architectural needs.

Horizontal composition fully integrates two networks into one. In this case, composition only occurs during the initial setup phase of the network integration – afterwards, the composed *TurfNet* operates exactly as a monolithic *TurfNet* would. In other words, composition does not visibly affect performance, scalability, security or other network properties.

Horizontal composition of networks allows integration of networks belonging to the same administration or different administrations. However, horizontally composing networks must all agree on the same address space, address allocation scheme and require a common routing mechanism. One example of horizontal composition occurs between different networks of a single or multiple cooperating network providers. However, composition between a service provider network and its customers' networks requires a different type of composition due to lack of trust and the desire to preserve some level of autonomy between the different parties. Consequently, a more loosely coupled form of composition is needed.

Vertical composition provides this looser form of composition. It enables independent networks with different network architectures that may belong to separate administrations to compose in a way that preserves their individual autonomy and specific internal operation. The gateway nodes, which are configured through the *TurfControls* involved in the composition process, enable the integration and interoperation of the otherwise fully independent networks. The overhead associated with this loose type of composition is acceptable in cases where closer composition is not an option due to administrative concerns, *e.g.*, lack of trust or desire for autonomy).

In terms of the *TurfNet* architecture, the existing Internet topology can be interpreted as consisting of horizontally composed networks, each being its own administrative domain or autonomous system. They share a common address space together with common routing and name resolution functions. The *TurfNet* architecture enables composition of this Internet-wide *TurfNet* with new access networks that feature different control functions, or for example, with vehicle area networks (VANs) that use their own non-IP communication infrastructure, Bluetooth-based personal area networks (PANs), or *ad hoc* networks. All of these networks temporarily or permanently compose with the current Internet through *TurfNet* gateways. The only constraint is that a common, global name space for all of them needs to be place.

Due to space limitations, the remainder of this paper focuses on the – arguably more interesting – vertical *TurfNet* composition, even though both types of composition are equally important pieces of the overall architecture.

4 Basic Operation

End-to-end communication across *TurfNet* boundaries is not trivial due to isolation and heterogeneity of the individual networks. *TurfNet* adopts the decoupling of names and locators from FARA's abstract architectural model [2]. It therefore uses names as global identifiers of *TurfNodes* that are different from the node addresses used for routing.

Because *TurfNode* addresses may not have end-to-end significance (they might merely be transient routing tags for *Turf*-local routing), the architecture uses the name registration and resolution process to find and setup the high-level routing path across composed *TurfNets*. End-to-end communication across *TurfNet* boundaries is thus a product of the following processes: *address allocation, node registration*, *name resolution* and *packet relaying*.

4.1 Turf-Local Address Allocation

A *TurfNode* joining a *TurfNet* first needs to obtain a *Turf*-local network address using the respective address allocation function of the *TurfControl* (*e.g.*, DHCP [13] or IPv6 auto-configuration [14]). This address only needs to be valid and meaningful within the local *Turf*. How individual *TurfNets* handle address assignment does not affect the inter-*Turf* architecture. Ideally, this process of address allocation should happen in a fully automated way.

4.2 Node Registration

Because individual *TurfNets* may have completely independent network addresses spaces, *TurfNodes* may not be directly addressable from outside their local *Turf* (similar to today's NAT'ed hosts). The lack of a global address space across all *TurfNets* prevents an external node (of another *TurfNet*) from addressing a local node directly. The *TurfNet* architecture exacerbates this problem, because different *TurfNets* may not only have overlapping but also completely different address spaces or network protocols (*e.g.,* IPv4, IPv6, or any other internetworking protocol).

Although NATs are often held responsible for breaking the end-to-end semantics of the Internet – and rightfully so – their hiding capabilities are a central feature of the *TurfNet* architecture. However, *TurfNet* carefully eliminates the disadvantages of NATs by introducing names as explicit, end-to-end node identifiers. Gateways are free to translate addresses and protocols when higher-layer entities bind to static names. Hiding *TurfNet* internals facilitates strong autonomy between *TurfNets* and minimizes shared global state. The ability to hide the internals of networks and the resulting autonomy is becoming critically important in today's commercial Internet, where even companies that own sufficient address space use NATs to control outside visibility of internals of their local networks.

A *TurfNode* that wants to be reachable for nodes outside of its local *Turf* registers its local address with the name resolution service of the local *TurfControl* (step 1 in Figure 5). To achieve reachability from external *TurfNodes*, this registration propagates up the hierarchy of composed *TurfNets* (step 2) via the *TurfControl* of the local *TurfNet*. Note that a *TurfNode* itself cannot register its address in external *TurfNets*, because it is not directly aware of their existence.

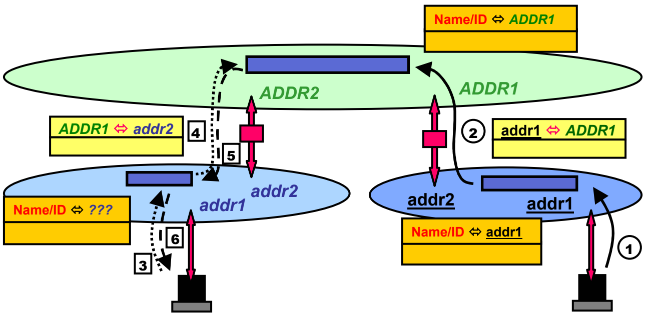

Figure 5. Node registration and name resolution across *TurfNets* with independent address spaces.

If a child and a parent *TurfNet* use separate or even different address spaces, a straightforward propagation of the address registration to the parent *TurfNet* would fail, because the addresses of the registered *TurfNode* have no meaning outside its local *TurfNet*. Consequently, a new *proxy address* for such a node needs to be allocated and registered with the parent *TurfNet*. This process may cause a node to receive different local proxy addresses in each *TurfNet* along the path. The gateways initiate the allocation of proxy addresses. For each address registration propagated to a parent *TurfControl*, the gateway that connects the *TurfNets* requests a new local network address from the address allocation function of the parent *TurfControl.* It then creates the necessary address/protocol translation state between the different addresses. The address allocation function operates in a distributed fashion; for example, each gateway might control its own pool of addresses.

Mappings from names to proxy addresses are *soft state* that times out if not refreshed periodically. This conforms to the requirement of using soft state between *Turf* boundaries. Using soft state for address mappings can also significantly reduce the necessary state in the gateways, because they maintain only the state associated with active nodes at any given time.

4.3 Name Resolution

If a *TurfNode* wants to communicate with a peer node, it requests name resolution through the *Turf*-local name resolution service. If the peer node is part of the same *TurfNet*, this is a fully local operation, as illustrated by steps 3 and 6 in Figure 5.

However, if the peer node is not part of the local *TurfNet*, the local name resolution function propagates the lookup to its parent *TurfNet(s)*, which then try to resolve the name within their respective domains (step 4). Because addresses may only be valid within a single *Turf*, the addresses that name resolution returns may differ from *TurfNet* to *TurfNet*. Thus, if the child and parent *TurfNet* use a different address spaces, gateways must allocate a new proxy address for the resolved name after a successful name resolution by a parent *TurfNet.*

When gateways must allocate a local proxy address (because the child and parent *TurfNets* use different address spaces), they also install the necessary address/protocol translation state for mapping between the different address spaces and/or network protocols. The gateway that receives the successful reply from the parent *TurfNet* can either perform this operation "in-band" or the local *TurfControl* can perform this operation "out-of-band" after it receives the name resolution response.

This address/protocol translation state is also *soft state*, in order to reduce gateway state when only a few inter-*Turf* communications are active. The specific conditions for flushing soft state depend on the individual types of communication. For example, exceeding the maximum idle time of an address translation entry could invalidate the soft state mapping between proxy addresses names.

It is a characteristic of the *TurfNet* architecture that both address registration and name resolution may include address allocation and creation of address/protocol translation state.

4.4 Packet Relaying

End-to-end communication among *TurfNodes* can begin as soon as the address resolution process completes successfully. If both communicating peers belong to the same *TurfNet*, their communication is a completely local process that does not involve inter-*Turf* mechanisms.

If the communicating peers belong to different *TurfNets*, packet relaying involves the following steps.

First, if the peering *TurfNets* use the same network address space and communication protocols, the gateway nodes merely act as traditional routers and forward traffic between the different administrative domains. The *Turf*-local routing protocols then connect (across the *Turf* boundaries) to facilitate inter-*Turf* routing in much the same way as today's inter-domain routing protocols do (*e.g.*, BGP).

Second, if the peering *TurfNets* use independent address spaces and/or different network protocols, *Turf* gateways also perform the necessary address and protocol translations when relaying packets. In this case, the address registration and name resolution procedures have already established the necessary proxy addresses and network address translation state at the gateway nodes along the communication path. Similar to today's Internet NATs, a *TurfNet* gateway adds a dynamic address translation rule for the reverse direction when a local *TurfNode* initiates communication with an external node. For this to work, a *TurfNet* gateway has to maintain network address translation state in both directions and must perform address translation on both source *and* destination addresses. This is often referred to as *twice-NAT* [15].

The result is that communication across *TurfNets* must always follow symmetric paths, because only those gateways have the necessary translation state. This is a change from the current Internet, which supports unidirectional links that cause forward and reverse traffic between two nodes to follow different, asymmetric paths. In reality, however, many Internet protocols and services do not deal with asymmetric paths well, especially if their characteristics (*e.g.,* bandwidth or delay) are sufficiently different. In the future, asymmetric paths across *TurfNets* could be supported by coordinating translation state across different sets of gateways for the forward and reverse path. The management and security aspects of such approaches are currently not well understood and the present *TurfNet* architecture consequently limits itself to supporting symmetric paths only.

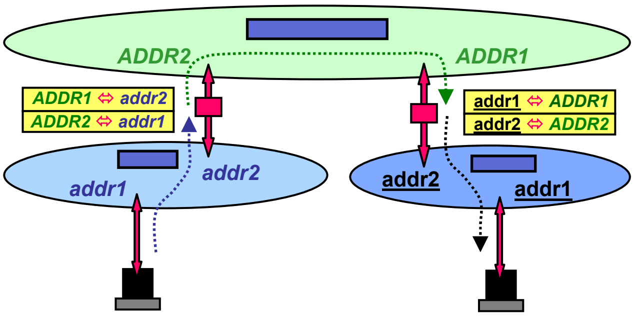

Figure 6. NAT-based packet relaying in case of peering *TurfNets* with independent address spaces and/or protocols.

Figure 6 illustrates packet relaying in detail. Note that although the example shown there is a simple two-stage scenario, the basic mechanism extends recursively to multiple stages. In the left of Figure 6, the initiator of the communication has address **addr1** obtains an external proxy address ***ADDR2*** in its parent *TurfNet* and creates an entry in the address translation table that maps between those addresses: ***ADDR2***⇔**addr1**. Note that different fonts are used for different address spaces: lower-case letters for the lower left, underlined letters for the lower right, and upper-case letters for the top *TurfNet*.

Figure 6 also illustrates the establishment of dynamic address translation state at the *TurfNet* gateways for reverse communication. Each gateway along the path must translate the source address of the sender. For this to work, these gateways must also establish a proxy address for the sender in every *TurfNet* along the transmission path. For the example in Figure 6, this means that the gateways first translate the sender's address (**addr1**) into proxy address ***ADDR2*** at the top-level *TurfNet* and then into **<u>addr2</u>** at the receiver's *TurfNet*.

When this stage finishes, it has created all required proxy addresses and their corresponding address translation state. Bidirectional communication between the peers is now possible through straightforward address translations.

5 Scalability Considerations

One central assumption of the *TurfNet* architecture is that no hard state exists between individual *TurfNets*; consequently, gateways and name resolution services may only use soft state. Especially critical for scalability and performance is the efficient storage of name-to-address bindings and network address translation state. Because the *TurfNet* architecture decouples domains, each *TurfNet* is free to choose an appropriate solution that satisfies its particular scalability and performance needs.

5.1 Namespaces and Name Resolution

One of the central ideas of the *TurfNet* architecture is the complete autonomy of individual *TurfNets*. They do not require global state, globally unique names or global identities for all communication entities. However, to reduce the problem of global naming to the problem of a global namespace (rather than to a globally distributed name service as in today's Internet), every *TurfNet* provides a local name resolver as a basic functionality of the *TurfControl*. The problem of assigning globally unique names is an orthogonal issue to be solved outside the *TurfNet* architecture.

Local name resolvers in every *TurfNet* that map globally unique names to *Turf*-local addresses allow each *TurfNet* to operate completely autonomous, without the need of external naming services.

In contrast to the Internet's Domain Name Service (DNS), name-to-address mappings in *TurfNet* are soft state, *i.e.*, they need to be updated regularly or they disappear automatically. Therefore, each `(name, address)` tuple has an associated *lifetime*. A *TurfNode* registers its *Turf*-local network addresses with the name service and must then periodically refresh these registrations to prevent them from timing out. Furthermore, `(name, address)` tuples have additional *cache* timers that are similar in function to DNS timers. They indicate how long clients may cache name-to-address resolutions before they must refresh them. A cache timer of zero indicates that clients must perform a new name resolution for each new transport-layer connection. For example, a cache time of zero can be used for mobile nodes that change locations.

Figure 5 illustrates inter-*TurfNet* communication, which relies on the fact that local *TurfNodes* that decide to be reachable from remote *TurfNodes* register their location with their parent *TurfNets*. This hierarchical name registration and resolution process ensures that a *TurfNode* that is part of any *TurfNet* in the overall composition can locate any registered node. If the *Turf*-local name service cannot resolve the address itself, the lookup request recursively propagates to the parent *TurfNets* until the name is resolved. Again, this recursive propagation is somewhat similar to the DNS.

A negative side effect of hierarchical name resolution is that top-level *TurfNets* require name-to-address mappings for all registered hosts, *i.e.,* all hosts that choose to be reachable by any node in the composed *TurfNets*. For example, if the Internet were to be rebuilt from composed *TurfNets*, the top-level *TurfNet* would require `(name, address)` tuples for any host that wants to be globally reachable. This example shows the importance of scalability considerations for large (composed) *TurfNets*.

An obvious approach to address the scalability issues is distributing the name-to-address resolution service across many servers. This approach can achieve a high degree of load balancing and fault tolerance. For example, a distributed hash table, such as Chord [16], FPN [17] or Koorde [18], could provide the necessary levels of fault-tolerance and performance for very large numbers of nodes.

The scalability of a name resolver depends entirely on the potential size of its *TurfNet* and the number of nodes that decide to be globally reachable. For example, small *TurfNets* such as an *ad hoc* access network or a Personal Area Networks (PAN) may simply implement a centralized resolver.

The *TurfNet* architecture hides the internal structures and control functions of *TurfNets* and consequently leaves the implementation characteristics of these services to the individual *TurfNets*. For example, whereas a personal-area *TurfNet* may implement the *TurfControl* as a centralized service on a single node, a larger composed *TurfNet* that potentially encompasses a worldwide network with billions of hosts must obviously choose a very different implementation approach to fulfill its specific scalability and performance demands.

5.2 Name Resolution Delegation

This section outlines a particular name resolution mechanism that supports the specific requirements of the *TurfNet* architecture and can scale up to large composed *TurfNets* with billions of *TurfNodes*. The proposed name resolution mechanism is of special interest for the *TurfNet* architecture as it supports highly dynamic address updates despite large-scale deployments (as for example needed for future mobile networks).

The main idea of the proposed mechanism is to split the name resolution process into two steps. The first step simply resolves the *Delegate Address Resolver (DlgAR)*, which is then responsible for the actual name-to-address resolution in a second step (see Figure 7).

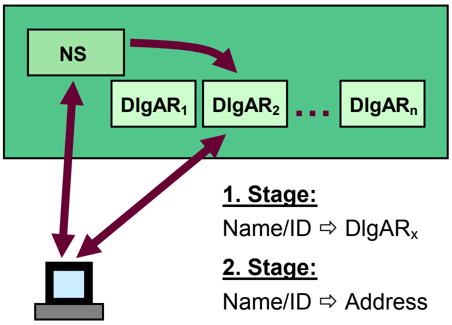

Figure 7. Delegate name resolution system.

Because the mapping from an actual name (or host identity) to the responsible *DlgAR* is expected to be relatively static, the result of this resolution is cacheable for long durations. For example, if this first mapping is based on the name (or host identity) prefix (*e.g.,* all names starting with “a…” map to *DlgAR$_x$, “www.ab...”* to *DlgAR$_y$*, etc.), repetition of the first lookup step will be very rarely needed. This achieves a high level of load balancing, because most lookups will only involve the responsible *DlgAR*s.

One benefit of this distribution mechanism is that only a single instance (or at the most a few instances, for reason of fault tolerance) maintains the actual name-to-address mapping. This enables highly dynamic changes, without the overhead of updating many servers. For example, a mobile node that changes its points of network attachment frequently could use this name resolution mechanism to handle the mobility management for new connections.

To further increase scalability of the proposed solution, one could extend the two-stage name lookup mechanism into a multi-stage name lookup mechanism.

5.3 Aggregation of Address Translation State in Gateways

Besides the scalability concerns of name resolution systems in large (composed) *TurfNets*, the architecture also has stringent scalability requirements for gateway nodes. For gateways to allow inter-*TurfNet* communication, they must perform address translations on all inbound and outbound packets. For large (composed) *TurfNets*, this requires the gateways to maintain a dynamic proxy address for any registered or active host that uses it. Besides proxy addresses, the gateway also has to hold the necessary address and/or protocol translation state.

Because a gateway that connects to a top-level (composed) *TurfNet* may provide address translation functionality for potentially huge numbers of nodes, minimizing required state is important. The amount of state required at *Turf* gateways is expected to be of a similar order of magnitude as in today's NAT gateways. Therefore, translation state is not expected to be a limiting factor for scalability.

Nevertheless, state aggregation can minimize state in *TurfNet* gateways. The basic idea is for gateways to allocate dynamic proxy addresses for nodes of sub-*TurfNets* in a way that aggregates addresses, such that one or at the most a few separate entries exist in the address translation table.

The following example illustrates how *TurfNets* can aggregate state. Without loss of generality, the example uses familiar IPv4 addresses. In this example, addresses only have to be unique within a single *TurfNet*. Even sub-*TurfNets* can use overlapping addresses. Figure 8 illustrates state aggregation in the case of an IPv4-based addressing scheme. The top-level *TurfNet* gateway on the left maintains only two aggregated proxy addresses (namely, 10.1.0.0/24 and 10.2.0.0/16) and the relevant address translation state for those sub-*TurfNets.*

This example also shows the advantage of aggregation for the address translation process. In the case of aggregated addresses (for example, the second entry in the address translation table of the top-level gateway: 10.2.0.0/16 ⇔ 10.10.0.0/16), the gateway only has to translate the prefix of the host addresses. In this particular example, only the class B prefix requires translation.

The prefix-based address translation method, based on subnet addresses rather than individual host addresses, has the advantage of only requiring a fraction of the regular address translation state. Prefix-based address translation can also reduce the processing overhead in gateway nodes, as only parts of the addresses must be changed. Particularly in gateway nodes of large composed networks, where address translation state may be highly aggregated, translations would affect only small parts of addresses.

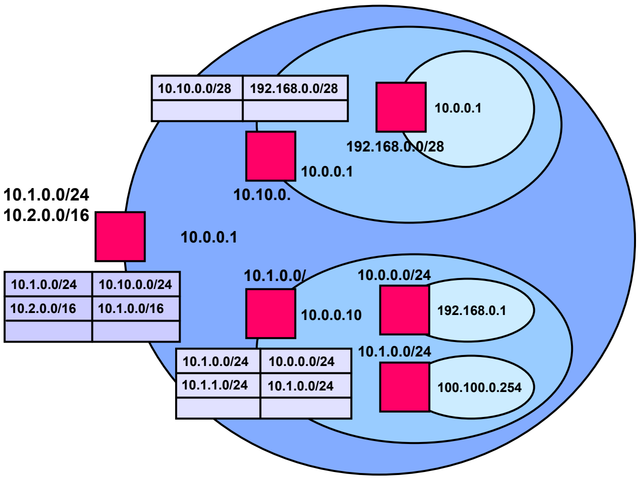

Figure 8. Aggregation of address translation state.

6 Architectural Evaluation

This section evaluates the proposed *TurfNet* architecture through a qualitative examination of the following aspects:

Scalability. The main bottlenecks of the architecture regarding scalability are the explicitly defined gateway nodes that relay inter-*TurfNet* traffic. The relay method – routing, address translation and/or protocol translation – can significantly affect *TurfNet* performance for large numbers of communication hosts and/or data flows. For example, when two core network providers compose their high-speed networks, performing address and/or protocol translation can be problematic. A better approach is to agree on a common address space and routing scheme. On the other hand, when a PAN network dynamically composes with a wireless hotspot network, composition of the different network types through network address translation may be an effective solution.

State aggregation mechanisms can mitigate the problem of state explosion in gateway nodes, as discussed above. However, another approach may be requires to address performance or load problems associated with address translation of all inter-*TurfNet* communication. One obvious approach is to introduce sufficient gateways to load-balance the necessary translation work. Note that the *TurfNet* architecture does not limit the number of gateways per *TurfNet*. Additional gateways will only add some extra control traffic.

Resilience. Inter-*TurfNet* communication relies on dedicated gateway nodes that are able to relay traffic between neighboring *TurfNets*. Hence, the architecture depends on the correct operation of those nodes. This dependency is similar to today's Internet, where NAT'ed networks also depend on the correct operation of the NAT gateway and the Internet at large depends on the correct operation of its routers.

One way to address this problem is to introduce sufficient backup gateways to allow failovers. However, designing a sufficiently fast failover mechanism may prove a challenge. Another approach to improve fault tolerance in the *TurfNet* architecture is through configuration of redundant gateway paths during the initial address registration phase and when resolving the addresses of a communication peer. Note that this could be done by the *TurfControls* along the inter-*Turf* path in a way that is completely transparent to the end nodes. The use of alternative addresses when establishing the end-to-end communication path to a *TurfNode* enables creation of disjoint paths. *TurfControls* along the inter-*Turf* communication path recursively choose different gateways, thereby creating disjoint high-level paths for alternative addresses.

Because communication based on redundant peer addresses flows over a different set of gateway nodes, failures of one gateway on the original path will not affect the others. This allows the communication initiator to switch to a peer's alternative address (thus using alternative paths) in the presence of a gateway failure. The fact that alternative paths through different gateways likely pass through different network service providers further increases resilience, because this approach circumvents problems that could affect whole provider networks.

Performance. The impact of the proposed *TurfNet* architecture on the overall networking performance must be considered carefully. The fact that all inter-*TurfNet* traffic passes through fixed gateway nodes introduces several potential network bottlenecks.

Today's Internet, in some respects, suffers from the same problem, because many home and corporate networks are located behind NAT boxes. In addition, most 2.5/3G mobile access networks relay all external communication through static NAT gateways. Nevertheless, the fact that today's Internet operates – despite the large number of middleboxes – illustrates that performance problems due to NATs are solvable. Correct provisioning of the gateway nodes, both in terms of performance and numbers, is important with an increasing volume of inter-*TurfNet* communication.

One way to address potential performance problems of *TurfNet* gateways due to extensive address translations is dedicated hardware support. Existing hardware solutions for high-speed "label switching" systems (*e.g.,* ATM, MPLS) could in the future also support fast address lookups and rewrites of source and destination addresses.

Flexibility. The key design objectives of the *TurfNet* architecture are creating administratively independent, autonomous networks domains and allowing their dynamic composition. *TurfNets* are fully self-contained and autonomous, even down to the type of addressing and/or the routing protocols they can use. This provides great flexibility for integration and composability of *TurfNets*. Clear administrative boundaries with minimal, but well-defined, control interfaces provide the basis for flexibility.

Mobility. Mobility support is another important criterion for network architectures, because more Internet nodes and subnetworks are expected to become mobile in the future. Due to the information-hiding capabilities of *TurfNet*, many *TurfNodes* may not be directly addressable. In this event, correspondent nodes will have to address the mobile node through its external proxy addresses. Because this proxy address may not change when the mobile device moves between different *TurfNets*, mobility management can be a local operation that is transparent to the correspondents. In case of a *Turf*-local handoff, the mobile node has to merely inform the local NAT gateway about its new internal address. The hierarchical structure of composed *TurfNets* allows such "local" handoffs at any level in the hierarchy. For example, if a mobile node moves between *TurfNets* that have a common parent *TurfNet*, the handoff only affects the parent *TurfNet*. This local handoff at the parent-level is completely transparent to those correspondent nodes that are located above its parent *TurfNet* or in any other sub-branch of the *TurfNet* hierarchy. Only correspondent nodes in the same branch of the hierarchy are affected by the move. To "repair" inter-*Turf* communication between those correspondent nodes and the mobile node after a handoff, the parent *TurfNet* (where the change of inter-*Turf* routing to the mobile takes pace) signals the respective gateways along the path to update the relevant relaying state. Note that the full specification of *TurfNet* mobility management procedure is currently still under investigation and therefore not yet included here.

7 Related Work

This section discusses related work that also aims at resolving problems of today's Internet architecture.

TRIAD [4] is a recently proposed Internet architecture that tries to resolve the lack of end-to-end connectivity of today's NAT'ed networks by means of an explicit content layer. Similar to the *TurfNet* architecture, TRIAD uses name identifiers rather than addresses for node identification and routing. Because network addresses in both architectures have no end-to-end significance (they are merely used as transient routing tags), both approaches rely on name lookup mechanisms to find and setup the high-level routing path across the independent network domains. However, the main difference between TRIAD and *TurfNet* lies in the way they handle high-level routing. Whereas TRIAD uses source-routing to forward packets, *TurfNet* uses the name registration and lookup mechanisms to configure high-level routing paths and their necessary address/protocol translation state. Another major difference is that TRIAD fully relies on IPv4 support in all transient network domains, whereas *TurfNet* can mask diverse addressing schemes and network protocols in transient network domains through the concept of proxy addresses and protocol translation at gateway nodes.

Plutarch [3] is another internetworking architecture that aims to subsume existing architectures like the Internet. Similar to *TurfNet*, the aim of Plutarch is to make the heterogeneity of existing and future networks explicit. To translate communication among heterogeneous network environments (*contexts*), Plutarch introduces the concepts of *interstitial functions*, which allow data to pass between two adjoining contexts. Nevertheless, Plutarch differs fundamentally from the *TurfNet* architecture with respect to naming and routing. Firstly, it assumes different namespaces per context, and secondly, routing is based on sender selection of a context chain and the configuration of the required interstitial functions in gateway nodes along the context chain.

The NAT-extended IP architecture IPNL [5] is another closely related approach. Like the *TurfNet* architecture, it also aims at truly isolating administratively independent IP subnetworks and domains by providing a mechanism to loosely integrate them. The proposed idea also uses NAT middleboxes to integrate networks with a potentially overlapping address space in a way that does not require renumbering. Two fundamental differences to the *TurfNet* architecture exist. First, it does not limit the number of hierarchical composition steps, whereas IPNL considers at the most two levels: NAT'ed *private realms* local networks and global *middle realms* networks. Second, the *TurfNet* architecture does not depend on specific addressing schemes and network protocols, but rather tries to provide a general solution that can integrate many approaches.

A technique similar to IPNL is proposed in 4+4 [19]. Here, address translation occurs also between private and public realms, although in this case it is envisaged to support several "middle" realms. In comparison to IPNL, 4+4 is simpler and allows incremental deployment in today's networks. A fundamental difference between the *TurfNet* architecture, and IPNL and 4+4 is that both architectures rely on globally unique host addresses, which consists of the concatenation of a node's public/global and private/local addresses.

Another related work is the Address Virtualization Enabling Service (AVES) [20]. The key idea here is to virtualize non-IP hosts or hosts that are not globally routable through so called waypoints. The waypoints then act as relays between standard IP hosts and those typically not addressable/reachable hosts. In that sense, AVES is very specific as it only tries to provide bi-directional connectivity for individual hosts. The real overlap between *TurfNet* and AVES lies in the way the waypoint relays are selected. Similar to *TurfNet*, non-IP hosts are dynamically bound to waypoints during the name resolution in a *connection-initiator-specific* fashion.

The concept of "network pointers" proposed for SelNet [21] is another related approach. Instead of using standard network addresses within data packets, SelNet introduces so called *selectors*, which allow network pointers (packet handlers) to change the processing semantics of packets as they traverse the network. As a result, SelNet requires a specialized routing protocol that allows mapping of routing information onto selectors.

This section has shown that *TurfNet* is in many aspects related to recent architectural proposals. However, the ability to compose completely diverse autonomous networks and the resultant benefits are a fundamentally new feature of the *TurfNet* architecture. The fact that *TurfNets* can fully mask difference within individual network domains is especially important in the light of growing tussles in cyberspace.

8 Conclusion and Future Work

This paper introduces the *TurfNet* architecture for global, packet-switched internetworks. The architecture addresses the challenges of deploying networks in competitive environments that require means for autonomous control and information hiding.

The *TurfNet* architecture supports horizontal and vertical composition. Vertical composition preserves full autonomy of composed *TurfNets* and supports integration of heterogeneous packet-based networks that use different, non-compatible network-level protocols. The use of a high-level control interface that only requires a common data format across all autonomous network domains allows control of border gateways that perform the required protocol and address translation for communication across *TurfNet* boundaries.

The autonomy and flexibility provided by *TurfNets* requires the use of soft state. The paper outlines methods that assure a high scalability of the architecture. An important aspect of the *TurfNet* architecture is the high-level or inter-*Turf* routing. This paper discussed the feasibility and realization of inter-*Turf* communication (*i.e.*, packet processing and forwarding), but did not yet specify a specific routing mechanism. Consequently, one area of future work lies in inter-*Turf* routing mechanisms that account for dynamic composition of (moving) networks, as well as performance and reliability for individual end-to-end communication.

Another aspect of the *TurfNet* architecture that requires further consideration is the introduction of virtual "overlay" *TurfNets*. Such virtual *Turfs* can integrate service-specific functionality into the network without complicating its basic functionality.

Finally, another focus of future work lies in evaluating the *TurfNet* architecture through simulations of its scalability properties; especially for large, composed networks. One important aspect of this work is the implementation and measurements of specific prefix-based address translation mechanisms.

Acknowledgements

This document is a byproduct of the *Ambient Networks* project, partially funded by the European Commission under its *Sixth Framework Programme*. It is provided "as is" and without any express or implied warranties, including, without limitation, the implied warranties of fitness for a particular purpose. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the *Ambient Networks* project or the European Commission.

References

1. D. Clark, J. Wroclawski, K. R. Sollins and R. Braden. Tussle in Cyberspace: Defining Tomorrow's Internet. Proc. ACM SIGCOMM 2002, Pittsburgh, August 2002, pp. 347-356.
2. D. Clark, R. Braden, A. Falk and V. Pingali. FARA: Reorganizing the Addressing Architecture. Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture, Germany, August 2003, pp. 313-321.
3. J. Crowcroft, S. Hand, R. Mortier, T. Roscoe and A. Warfield. Plutarch: An Argument for Network Pluralism. Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture, Germany, August 2003, pp. 258-266.
4. D. R. Cheriton and M. Gritter. TRIAD: A Scalable Deployable NAT-based Internet Architecture. Stanford Computer Science Technical Report, January 2000.
5. P. Francis and R. Gummadi. IPNL: A NAT-Extended Internet Architecture. Proc. ACM SIGCOMM, San Diego, CA, USA, August 2001, pp.69-80.

6. K. R. Sollins. Designing for Scale and Differentiation. Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture, Germany, August 2003, pp. 267-276.
7. P. Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, November 1987.
8. R. Moskowitz and P. Nikander. Host Identity Protocol Architecture. Work in Progress (draft-moskowitz-hip-arch-06.txt), June 2004.
9. H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica and M. Walfish. A Layered Naming Architecture for the Internet. To appear Proc. ACM SIGCOMM, Portland, OR, USA, August 2004.
10. R. Braden, D. Clark, S. Shenker and J. Wroclawski. Developing a Next-Generation Internet Architecture, July 2000. Whitepaper, available at http://www.isi.edu/newarch/DOCUMENTS/WhitePaper.ps.
11. N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer and H. Karl. Ambient Networks - An Architecture for Communication Networks Beyond 3G. IEEE Wireless Communications, Vol. 11, No. 2, April 2004, pp.14-22.
12. C. Kappler, P. Mendes, C. Prehofer, P. Pöyhönen and D. Zhou. A Framework for Self-organized Network Composition. Proc. 1st International Workshop on Autonomic Communication (WAC 2004), Berlin, Germany, October 2004.
13. R. Droms. Dynamic Host Configuration Protocol. RFC 2131, March 1997.
14. S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.
15. P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, August 1999.
16. I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. IEEE/ACM Transactions on Networking, Vol. 11, No. 1, February 2003, pp. 17-32.
17. C. Dubnicki, C. Ungureanu and W. Kilian. FPN: A Distributed Hash Table for Commercial Applications. Proc. HPDC-13, Honolulu, HI, USA, June 2004.
18. M. F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. Proc. 2nd Workshop on Peer-to-Peer Systems, Berkeley, CA, February 2003, pp. 98-107.
20. Z. Turanyi, A. Valko and A. Campbell. 4+4: An Architecture for Evolving the Internet Address Space Back Towards Transparency. *ACM SIGCOMM Computer Communication Review*, Vol. 33, No 5, October 2003, pp 43-54.
21. T.S.E. Ng, I. Stoica and H. Zhang. A Waypoint Service Approach to Connect Heterogeneous Internet Address Spaces. Proc. USENIX Annual Technical Conference, Boston, MA, USA, June 2001, pp. 319-332.
22. Christian Tschudin and Richard Gold. Network Pointers. Proc. 1st Workshop on Hot Topics in Networks (HotNets-I), Princeton, New Jersey, October 2002.