Plutarch: An Argument for Network Pluralism

Jon Crowcroft[*], Steven Hand[*], Richard Mortier[†], Timothy Roscoe[‡], Andrew Warfield[*]

March 24, 2003

Abstract

It is widely accepted that the current Internet architecture is insufficient for the future: problems such as address space scarcity, mobility and non-universal connectivity are already with us, and stand to be exacerbated by the explosion of wireless, ad-hoc and sensor networks. Furthermore, it is far from clear that the ubiquitous use of standard transport and name resolution protocols will remain practicable or even desirable.

In this paper we propose *Plutarch,* a new inter-networking architecture. It subsumes existing architectures such as that defined by the Internet Protocol suite, but makes explicit the heterogeneity that contemporary inter-networking schemes attempt to mask. To handle this heterogeneity, we introduce the notions of *context* and *interstitial function*, and describe a supporting architecture. We discuss the benefits, present some potential scenarios, and consider the research challenges posed.

1 Introduction

The remarkable success of the global Internet is frequently attributed to a set of design decisions that prioritize simplicity and robustness through a strongly specified suite of protocols, and the incorporation of only essential mechanisms within the network itself. The astounding growth of this network over the past thirty years serves as a clear testament to the wisdom of these principles.

However, in recent years, the Internet's architectural assumptions have been fundamentally challenged. In particular, the introduction of specialized networks such as sensor networks, along with various middleboxes have all begun to strain the existing framework. As a result, we propose *Plutarch,* a new framework for next generation networks. It differs from the existing Internet architecture primarily in that it embraces heterogeneity in the hope of allowing radical innovation. The homogeneous Internet architecture and its advantages are not abandoned but retained as one architecture among many.

In Plutarch we divide the world into *contexts*, each comprising some set of hosts, routers, switches, network links and so forth. Within a context we expect homogeneity regarding such things as addresses, packet formats, transport protocols and naming services. Distinct contexts differ in at least one of these areas.

Communication across a set of contexts is enabled by *interstitial functions*, which map between the sets of functionalities encapsulated by contexts. We can divide such functionalities into four main areas:

Addressing: Mapping between different address contexts is a well-understood function (one that exists today in NAT boxes, for example). We suggest that programmatic interfaces to such functionality should be exposed, allowing mappings to be automatically set up, maintained and managed by network users.

Naming: Today, DNS provides a single global namespace, with management handled by hierarchical delegation. We predict that emerging services such as VoIP and personal area networking will favour an alternative approach of mapping between a plurality of naming systems, for reasons of scalability and administrative overhead.

[*]University of Cambridge Computer Lab, UK.
[†]Microsoft Research Cambridge, UK
[‡]Intel Research, Berkeley, CA, USA

Routing: Different styles of routing protocol are appropriate in different networks. For example, connecting an ad-hoc wireless network with on-demand routing to an Internet AS running OSPF and BGP requires a more complex mapping than simply exposing the routes of each network via BGP. Using an explicit interstitial function to map routing information between these networks hides the churn of Internet routes from the on-demand protocol, deals with the wireless network protocol's broadcast, and hides the instability and mutability of routes within the wireless network.

Transport: A single transport protocol struggles to deal with all network technologies; symptomatic of this problem is the example of wireless-specific TCP implementations that use techniques such as splitting and proxying [9]. Optimizing transport protocols for specific network types has many advantages (for example, the use of large fat pipes in Grid infrastructures) but these come at the cost of working out when such optimizations are appropriate. Explicit interstitial functions provide well-defined points at which such decisions can be taken.

By not limiting the set of interstitial functions that may be defined and built we hope to support extensibility within existing networks, and encourage innovation within new ones.

2 Motivation

We present the Plutarch framework for a number of reasons. The principal one is the concrete problem of connecting networks where a common overlay protocol such as IPv4 or IPv6 is infeasible or undesirable, for example sensor networks, or specialized networks which offer valuable intra-domain functionality which IP must ignore. We discuss this issue, and how our solution addresses it, in the rest of this paper.

The second reason is that the abstract model underlying Plutarch captures the state of the Internet we see today better than models based on the Internet's original architectural principles [11]. This is not to say these principles are wrong or inappropriate; rather, we claim that a conceptual framework based upon them does not facilitate clear thinking about the future of the network. They may be the right principles, but they are not an adequate frame.

Finally, a model of networking based on explicit contexts provides a clearer framework within which to debate future architectural changes to the Internet than the current tradition allows. We hope it will allow the debate to move forward.

We accept that we might appear to be moving the debate back, to the days where the Internet was a *concatenation* of disparate networks (ARPANET, MILNET, SATNET, etc), or *catenet*, rather than the universal lowest-common-denominator overlay network it latterly became. However, we feel that there is merit in revisiting this approach to networking and extending it to take account of much more recent computer science and distributed systems research. One could view the approach of Plutarch as application of concepts such as late binding and context relative naming from the last two decades of distributed systems research [23].

Consequently, this paper concentrates on naming and addressing issues for establishing connectivity between radically heterogeneous networks, a problem that the Internet Protocol only partially solved. In this paper, we are literally concerned only with 'inter-networking,' and not with any of the many other networking issues such as the units, timeliness or guarantee of resource allocation, security or auditing. A concrete realization of our framework must address these issues, but within the contexts of the particular networks being connected: we do not believe it is sensible to address them through a single unifying overlay network protocol.

3 The Internet Protocol Model

The IP suite imposes a single networking model and addressing scheme over the many different underlying network types it supports. This model [6, 11] is characterized by independent datagrams, a single, global address space for all endpoints, and unreliable best-effort delivery. The resulting homogeneity has allowed both overlying applications and underlying

network implementations to be deployed independently, and has seen the network as a whole develop at a rapid rate. The homogeneity of the Internet at the IP level together with ease of implementation are usually cited as the principal reasons for the success of the Internet.

Although this homogeneity has provided the scaffolding to develop today's global network, it is now an inhibitor of further innovation. The IP suite forms a strict 'semantic bottleneck' to which it is increasingly difficult to cleanly incorporate anything but incremental modification. A prime example of an unclean modification that has nonetheless taken hold is Network Address Translation (NAT) [20]. NAT boxes serve a very useful purpose in the real Internet, while breaking one of the model's fundamental assumptions: all machines on the Internet are equal in terms of connectivity (if not capacity or bandwidth).

In fact, it can be argued that this assumption has been invalid for some time, regardless of the presence or absence of NAT. Due to the nature of Internet routing and the complexity of running a network (or any distributed system) of such scale, it is common to discover that two seemingly connected Internet nodes cannot communicate directly. This recently lead to the development of the Resilient Overlay Network (RON) [2].

However, we believe that this bottleneck is an inevitable consequence of the homogenization of the network layer, rather than a weakness inherent in IP. Although helpful for a number of years, while the network was undergoing its initial 'big bang' phase, the approximations in the model are now becoming too out-of-step with reality. Consequently, merely replacing the existing protocol suite will not be sufficient to avoid future similar shortcomings; the model itself must be addressed.

4 Architecture

In contrast to proposals for new protocols or modifications to existing ones, we suggest that future network architectures focus on mechanisms allowing inter-operation of many heterogeneous networks without mandating a one-size-fits-all protocol suite. The extensive heterogeneity of contemporary networks should be embraced.

More concretely, functions such as naming, addressing, routing and transport must be supported end-to-end across radically heterogeneous networks through the addition of suitable explicit interaction at boundaries. By making these regime transitions *explicit*, we believe that (*i*) the network model will more accurately reflect the network's reality, and (*ii*) the network model will be more extensible, allowing new services to more easily be incorporated at all architectural layers.

4.1 End-to-End naming and addressing

In our view, the twin functions of naming and addressing should be implemented in accordance with the end-to-end argument. Paradoxically, the current Internet imposes single mechanisms for addressing (IPv4, by design) and naming (DNS, by an accident of evolution) from the middle of the network. This leads to a requirement for globally-bound names and addresses. As we have seen, the model is already insufficient to capture the architecture of the current Internet (NATs, IPv4/v6 gateways, dynamic DNS servers, etc.), and does not address the connection of other, radically different networks (sensor nets, planetary-scale overlays, etc.).

Rather than attempting put the genie back in the bottle by imposing a single global IPv6 addressing scheme everywhere (already unrealistic in the face of simple devices such as sensors), we propose here a more heterogeneous naming scheme.

In this scheme there are no global names or addresses. Instead, each network end system exists in an explicit context[1] in which all names and addresses usable by the end system must be bound. This allows flexibility in end systems by removing the requirement for homogeneity imposed by the 'middle' of the IP network, and moving naming and addressing policy decisions towards the end systems. Passing a name or address from one such context to another

[1] IPv4 hosts today are observed to exist in a 'context of no context': the context certainly exists, but is not made explicit.

entails rebinding the referent of the name in the destination context. This operation is carried out by an *interstitial function*, described below, and is the key challenge in heterogeneous networking.

This model has two compelling features. First, it neatly captures the reality of IP networking today, and in particular the use of NATs, proxies, and similar. Second, it extends to future networking technologies while still encompassing the current Internet, and without sacrificing the factors which have made it successful as a technology.

For example, a large cluster of very small networked sensors (such as Berkeley motes) can send data to an Internet host even though they cannot implement an IP stack (due to computational, memory, and power limitations, and intermittent connectivity) by binding an address in their own local network context which corresponds to a gateway to the host. Conversely, every sensor can be addressed from an IPv6 network by projecting the sensor network address space onto a subset of the IPv6 address space.

The model is clearly not limited to this two-context case. Two sensor clouds (or two mobile phone networks) can use the Internet for transit by appropriate binding of names in contexts. Similarly, we can accomodate the present-day example of two IP networks routing data between them over a third network technology: the address bindings and translations in this case are realised in the NAT facilities at network boundaries.

The change in this case is one of emphasis as much as anything: rather than viewing non-IP (or non-IPv4, or non-globally-routable-IP) networks as exceptions peripheral to a central IP network, we advocate recognizing them as peer networks and addressing the end-to-end problems in communicating between such heterogeneous peers, of which the current Internet is but one.

The central problems in this scheme are communicating and resolving names and addresses across network boundaries. The goal of our architecture is to provide a set of compositional building blocks that may be used to allow the composition of heterogeneous networks in order to provide an end-to-end service. The two abstractions we propose are the *context* and the *interstitial function* (IF).

4.2 Contexts

Contexts serve two purposes: first, they describe communication mechanisms embodied by different networks, within which an endpoint might bind a particular communication session; second, they serve as descriptors allowing end-to-end services to be composed via the application of network closures.

Within the Plutarch system, communication takes place between endpoints within *contexts*. Following Saltzer's notion of 'context' [19] or the ANSA notion of 'naming context' [23], a context is abstractly a set of *bindings* with reference to which *names* may be resolved. In Plutarch, a context describes a region of the network that is homogeneous in some regard. All names, be they DNS names, IP addresses, Ethernet MAC addresses, users, network links, etc, are resolved within some context.

For instance, a context describing a local LAN environment may specify that the available link-layer protocol is Ethernet, and that the network supports link speeds up to 100 Mb/s. Alternatively, a context might describe the local administrative space corresponding roughly to the Autonomous System (AS) within which a machine with a given IP address resides.

An endpoint is likely to exist within multiple contexts simultaneously. An obvious example of this is a machine supporting both a 100 Mb/s Ethernet interface and an ATM interface: such a machine exists in two distinct contexts each representing the properties of communication across the different interfaces.

Context membership may be dynamic, and such dynamic contexts should provide suitable mechanisms for members joining and leaving. A context may border other contexts, and multiple nested sets of contexts may exist (i.e. there is no notion of a universal 'root context').

An example of nested contexts might be a machine with just a single 100 Mb/s interface but supporting an IP protocol stack. It exists simultaneously within the local Ethernet segment, but also the local IP LAN

and potentially the wider Internet. The context representing the local Ethernet can be viewed as a specialization of the enclosing IP LAN context, itself a specialization of the enclosing Internet context.

4.3 Interstitial Functions

In order to accommodate the differences between contexts, while still providing an end-to-end service, data may have to be manipulated at context borders. To achieve this goal, we introduce the *Interstitial Function* (IF), whose purpose is to allow data to pass between two adjoining contexts. Contemporary examples of IFs include NAT boxes, signaling gateways, and BGP routers. However, we also envisage situations where IFs may explicitly be used to bridge dissimilar transport networks (e.g. IPv4 onto ATM) or to provide high-level service modification, such as transcoding video streams or inserting forward error correction on unreliable links.

IFs logically bridge two contexts, and so are composed of two interfaces representing the two contexts, coupled with some internal mechanism for translating data arriving at one context into the other context. IFs can be used to form chains of contexts connecting endpoints; such a chain can itself be referred to as a context.

In situations where they make no decisions based on such knowledge, endpoints might not wish to know the details of all the contexts in the chain. For such endpoints, the properties of the context representing the chain might be restricted to the properties of the endpoint's own context (which might themselves be hidden from a suitably disinterested endpoint) plus the fact that the endpoint's communication partners can be reached. More interested endpoints might be made aware of multiple possible context chains to their communication partners, and make their own choice as to which context chain to use. This allows for application specific optimization where appropriate.

In general we expect IFs to require a certain amount of state; this will generally be soft-state, and may be acquired epidemically or as a side-effect of some user action. Note that even though some state may be considered to apply per-flow (e.g. an entry in a contemporary NAT forwarding table), this does not mean that it must be *established* per flow, or that any per-flow signalling is required.

5 Examples

In this section, we discuss concrete examples of how a context-based networking architecture works or would work in practice. Some of these examples are scenarios which are outside the scope of the Internet protocols, and consequently either currently infeasible, or handled by ad-hoc application-specific methods. Others are current networking practices which fit into our model unmodified. We start with the existing Internet itself.

5.1 The IPv4 Internet

In considering the pragmatic issues of deploying a new Internet architecture, our proposed approach has two strengths: first, the existing Internet may remain completely unchanged, and second, contexts may be deployed incrementally.

The contemporary, globally routeable Internet exists as a context in itself. Within this context, mechanisms remain unchanged and the network may continue to evolve as it always has: we do not advocate the deliberate replacement of the Internet, but the peaceful coexistence of the Internet as one context among many.

We already see examples of alternative contexts alongside the Internet, for instance NAT-connected LANs. In this case the interstitial function is performed by NAT boxes, and is simplified by the use of IP on both sides and the interstitial address mapping being (at present) relatively static.

With the context-supporting services deployed within the existing Internet, other contexts may be connected at its borders (sensor networks, NAT-connected LANs, etc.), above it as overlays, alongside it (emerging network protocols, of which IPv6 is the strongest candidate), or even beneath it (private link layer networks). These other networks may

innovate and evolve relatively independently, using IFs to interact with one another where necessary.

Note also that an initial deployment of small numbers of contexts will not require a large degree of support service layer within the Internet. Our current efforts are to provide a peer-to-peer overlay allowing border nodes hosting IFs to communicate with one another permitting adjoining contexts to interact. If successful we imagine that the load on this peer-to-peer network may require supporting by a more structured overlay.

5.2 Transiting Multiple Contexts

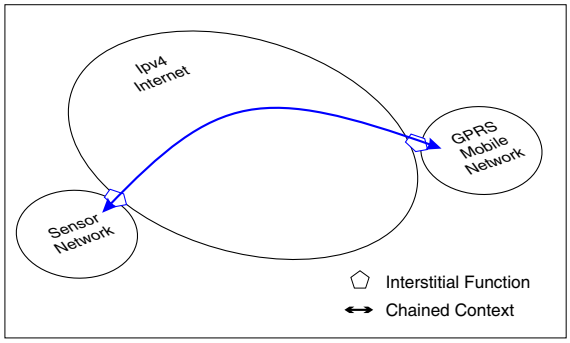

Figure 1: Connecting Across Contexts

Figure 1 shows a situation which might arise in today's network. A wayward graduate student, bound for a networking conference in Germany desires to query the state of his research sensor network, attached to the Internet through a host in Vancouver. His connectivity in Europe is through a GPRS phone, attached to his laptop. As the IP service provided by his GPRS provider is mapped through an opaque gateway, and the network sensors, in an effort to reduce power, do not even implement an IP stack, neither end of this connection is directly addressable from points within the Internet.

An end-to-end connection between the laptop and sensor network can be set up using our proposed architecture as follows:

Stage 1: Name resolution. As there are no global names, the first stage in connecting the hosts is to query for the desired context (or host, or service). Such queries take the form of a collection of name=value pairs and our present approach to a naming service is to employ epidemic-style gossip [3] advertisements and queries across contexts. The route query describes the target name, and the properties of the communication channel to be provided. For instance:

route(name=myExperimentalSensorNetwork,
props=(protocol=QueryProtocolv1.2,
transit(connection=reliableByteStream)))

Note that the target name need not have any meaning outside of the target context: it is simply an identifier used for searching. The transit parameters describe properties of any midpoint contexts involved in the link. These may be taken as hints to constrain the distributed search, and validated by the querier on considering the result set. Note also that in this simple example, we are connecting a set of specific end-to-end hosts. The query model is intended to support other approaches to location, such as Intentional Naming [1].

Queries percolate through the lookup fabric and result in a set of candidate replies being returned to the requesting host. Replies are in the form of chained-context descriptors. These list a vector of context descriptions and the associated IFs that lie between them.

In our example, the service and protocol being requested are very specific. Only two replies are received on the student's laptop, describing a context chain crossing the Internet, but requiring interstitial functions as shown in Figure 1. The variation in the set of replies indicate that the local GPRS context provides both plain IP, and a modified IP, incorporating forward error correction to help survive packet loss.

Stage 2: Chained context instantiation. Some logic in the host selects one of these context chains to form the connection. The appropriate query result is turned into an instantiation request and forwarded to the appropriate border node in the GPRS context where an IF is configured. The request is then forwarded along to the border of the sensor network, where a second IF is configured. As joining some contexts (configuring suitable IFs) may require some form of authentication, challenges may be is-

sued back along the partially open context chain. Alternatively, this process may be short circuited by embedding authentication in the initial request message.

Once the chained context has been instantiated, it is added to the laptop's list of known contexts. A set of bindings is installed between the new context name and the related communications mechanisms on the laptop. In addition, the context is cached with the name service to facilitate future connections.

Stage 3: Communications. Once the context has been bound to the local host, applications may interact with it through the associated mechanisms. In many cases these bindings are likely to involve patching a new context below the socket layer, but other approaches are also worth considering.

5.3 Other Examples

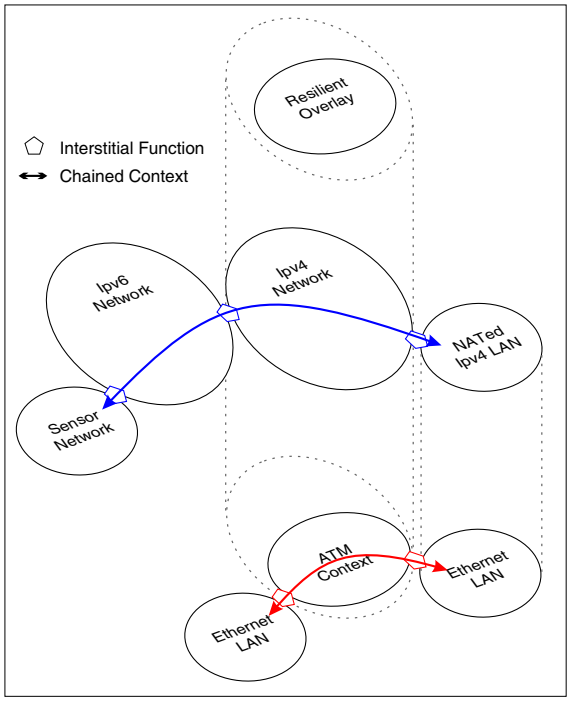

Figure 2: A Variety of Contexts

Figure 2 shows a set of contexts from which additional examples may be drawn. The figure shows contexts with a variety of relationships to one another, and two examples of chained contexts.

The IPv4 and IPv6 networks are shown alongside one another, as they they provide parallel but incompatible network service. Note that this diagram is not meant to imply that participation in these two contexts is mutually exclusive. On the contrary, there may be a great deal of overlap in the memberships. The separate contexts simply show the boundaries of homogeneous protocols.

The vertical set of contexts associated with the IPv4 context is to show the relationship between context membership. The resilient overlay context has been constructed above the IPv4 context. It provides an additional service, but is constructed using IPv4 as an underlying communications mechanism.

The chained context between the sensor network and the NATed LAN is very similar to the example presented above. In this case, there is the addition of the IPv6 context in the chain, resulting in an additional interstitial function.

In the second chained context, two Ethernet LANs are connected across an ATM network. Here the ATM network contains a set of nodes also participating in the IPv4 network[2]. In this case, the context chain is meant to indicate that a link-layer bridge has been implemented across the two interstitial functions [15].

6 Some Strawmen

To add more solid fuel to the discussion, this section briefly describes strawman interfaces to three components of this system: the context, the IF and the Plutarch management service.

Context Interface

```
inspect() → value
list() → [name]*
insert(name, value)
lookup(name) → value
remove(name)
```

The *context* interface is to be used at endsystems

[2] Although this is by no means a constraint—additional nodes could be involved.

to interact with particular instances of contexts. A contemporary analogy might be the Dynamic DNS API [24].

inspect() returns information pertinent to communication bound within this context;

list() returns the list of names in this context;

insert() adds a value to this context keyed by a name, replacing any value that already exists under name in this context; values may themselves be lists, but the semantics of merging multiple values under a given name is context dependent and to be handled by some entity subject to suitable access permissions;

lookup() translates a name to a value within this context;

remove() removes the entry keyed by name from this context;

Interstitial Function Interface

```
inspect() → (ctxt1, ctxt2, value)
configure(value, caps)
```

The *interstitial function* interface is to be used to allow endsystems to interact with IFs. Most existing IFs do not have programmatic interfaces, but an approximation might be provided by the various firewall and NAT coordination protocols [].

inspect() discovers the two contexts that a particular IF connects, along with any other relevant information;

configure() configures an existing interstitial function in some way appropriate to the particular IF subject to the capabilities presented.

Plutarch Management Service Interface

```
register(props) → ctxt
deregister(ctxt)
link(ctxt1, ctxt2) → [ifun]*
lookup(props, ttl, caps) → [ctxt]*
route(name, props,
      ttl, caps) → [ctxt-chain]*
```

The *Plutarch management service* interface[3] gives access to a distributed service formed of multiple cooperating instances that may be controlled by distinct administrations. In keeping with the spirit of Plutarch, there may be many implementations and instances of Plutarchies interacting via suitable interstitial functions; we believe that implementations will have to support functionality similar to that shown here. Management of capabilities to support a particular Plutarchy is not specified within the architecture.

register() register a context with the specified communication properties;

deregister() deregisters a context from this Plutarchy;

link() returns a list of potentially unconfigured IFs existing between two contexts;

lookup() recovers the contexts supporting particular properties, subject to a hop limit on the propagation of the query and capabilities presented in the request;

route() discover a context chain or chains limited in length by ttl, containing contexts accessible according to the capabilities presented, and supporting particular properties terminating with a context in which the given name is known.

link(), *lookup()*, and *route()* should all support delegation both between components of this Plutarchy and between instances of Plutarchies. As such, their implementations should probably return results asynchronously, or at least support timeouts of some form. Names may be registered in multiple contexts and are opaque in contexts in which they are not registered, so endsystems must implement some application or platform specific policy to select appropriately from the results of a *lookup()* or *route()*.

As already noted, a *context chain* is a sequence of contexts interleaved with IFs. Endsystems may wish to know and specify the details of a context chain, in which case should an element of the chain fail, the endsystem must be notified that it should deal with this, perhaps by invoking another *route()* operation. However, as part of the properties pass to the *route()* operation, an endsystem may specify the detail it requires in responses and whether repair of a broken chain should be automatic. In this way, the Plutarchy

[3] Referred to hereafter as the *Plutarchy* for brevity.

can ensure that route repair occurs in much the same way as occurs with current Internet routing.

7 Related Work

We are by no means the only researchers advocating the necessity of a new architecture: indeed, the zeitgeist appears to firmly embrace the notion, as witnessed by this very workshop. Much of this work has been inspired by the perceived failure of the IETF IPng working group.

One of the best known efforts is the ambitious NewArch project being carried out between ISI, MIT and ICSI [4]. This hopes to present a detailed blueprint for next generation networking, addressing topics such as mobility, quality of service and interplanetary communication while ideally retaining the best of the original architectural principles. One concrete proposal from this team is the notion of a *role-based architecture* in which layering is eschewed in order to gain maximum flexibility [5].

The uniformity of the approach is elegant although perhaps unnecessary, and the suggestion that IPv4 be retained as a base layer does not readily support sensor networks or scalable multicast, for example. Nonetheless the basic scheme of explicitly communicating certain networking semantics could be usefully applied to our interstitial functions.

Another recent next-generation architecture proposal is Triad [10, 14]. Much like Plutarch, Triad replaces traditional name lookup with something more akin to searching. However their focus is on content-based naming and routing rather than semi-structured data queries; while novel, it is not clear that content is king in all contexts. Their deployment model supposes the existing global IPv4 network augmented with 'WRAP' gateways to allow communication between addressing realms.

The general notion of providing translation between IPv4 realms to avoid perceived problems with NAT boxes was proposed independently as IP Next Layer (IPNL) [13]. IPNL separates the communication path into three: an originating *private realm*, a global *middle realm* and a second terminating private realm. These realms are similar to our notion of contexts, but are more limited in type and function.

A technique similar to although simpler than IPNL is proposed in 4+4 [22]. Once again, address translation occurs between private and more widely known realms, although in this case the authors envisage more than one middle realm. The 4+4 scheme is simple, elegant and incrementally deployable, but it limits itself to network-layer issues and does not propose new naming or transport-layer functionality.

Another recent scheme, AVES [17], specifically targets the problem of non-IP hosts, including hosts within an addressing realm which reuses IP addresses. Their key notion is to virtualize these non-IP hosts by using *waypoints*: globally addressable middle boxes [7] which act as relays for IP traffic. Once more the focus is on IP connectivity and not on a new architecture *per se*.

Our proposal of multiple contexts which explicitly interwork is perhaps most reminiscent of the idea of the Metanet [25]. In this architecture, the network is divided into *regions*, each of which models a particular real-world set of requirements and limitations; a set of *waypoints* exist at boundaries between regions and perform translation as needed. Our work is based on the same general notion, but attempts to explore further how such a scheme could be made to work.

8 Conclusion and Future Directions

We have presented Plutarch, an inter-networking architecture that makes heterogeneity explicit so that it may be exploited. We believe that this better represents the status quo in the Internet and more importantly, is extensible enough to capture future network evolution. Note that, as with most architectures, this paper is really just a starting point: considerable future work is required in order to produce an complete design and implementation reflecting this architecture.

However, there is an existing body of work on which we can draw. For example, the Nimrod routing architecture [8] with its 'pull' mode of operation seems well suited to our asynchronous scheme for learn-

ing of context chains, although clearly modification would be required to handle query-based route finding. In in terms of communication models, the Internet Indirection Infrastructure [21] provides good evidence for the power of making packet forwarding a sometimes application-level task. We also hope to learn from existing work on active networks and protocol composition [12, 16, 18], although we do not particularly espouse a 'pure' active networking approach.

Notwithstanding this body of work, substantial research is introduced by this architecture. Issues untouched in this paper include scalable mechanisms for inter-context routing, IF discovery and failure notification; policy issues in choice of context and context-chain; and mechanisms to expose these things to the programmer, either through an existing network API such as the sockets API, or through some extended or new API. Transport protocols that can optimize for the context or contexts over which they operate is also worth investigation: related work here includes techniques such as TCP splicing, where a TCP connection is severed at a proxy between, for example, a GPRS network and the wired Internet, for performance reasons (essentially to overcome the performance mismatch between the two networks).

Mechanisms to alleviate some of the current strain on management of the Internet and Internet address allocation might also become feasible through the application of contexts and explicit boundaries: there are many anecdotes[4] suggesting that the current structures for allocation of global IP address blocks, and notification of new allocations are inadequate, for example. We believe that avoiding the need for a single global management entity (or hierarchy of entities) may allow techniques from peer-to-peer networking, and other highly scalable systems to be usefully applied.

References

[1] ADJIE-WINOTO, W., SCHWARTZ, E., BALAKRISHNAN, H., AND LILLEY, J. The Design and Implementation of an Intentional Naming System. In *Proceedings of the 17th ACM SIGOPS Symposium on Operating Systems Principles* (December 1999), pp. 186–201.

[2] ANDERSEN, D. G., BALAKRISHNAN, H., KAASHOEK, M. F., AND MORRIS, R. Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP'01), Banff, Canada* (October 2001).

[3] BIRMAN, K. The Surprising Power of Epidemic Communication. In *Proceedings, Workshop on Future Directions in Distributed Computing (FuDiCo 2002). Bertinoro, Italy* (June 2002), Spinger-Verlag.

[4] BRADEN, R., CLARK, D., SHENKER, S., AND WROCLAWSKI, J. Developing a Next-Generation Internet Architecture, July 2002. Whitepaper, available at `http://www.isi.edu/newarch/DOCUMENTS/WhitePaper.ps`.

[5] BRADEN, R., FABER, T., AND HANDLEY, M. From Protocol Stack to Protocol Heap — Role-Based Architecture. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)* (October 2002).

[6] BUSH, R., AND MEYER, D. Some Internet Architectural Guidelines and Philosophy. RFC 3439, IETF, Dec. 2002.

[7] CARPENTER, B., AND BRIM, S. Middleboxes: Taxonomy and Issues. RFC 3234, IETF, Feb. 2002.

[8] CASTINEYRA, I., CHIAPPA, N., AND STEENSTRUP, M. The Nimrod Routing Architecture. RFC 1992, IETF, Aug. 1996.

[9] CHAKRAVORTY, R., CARTWRIGHT, J., AND PRATT, I. Practical experience with TCP over GPRS. In *Proceedings of IEEE GLOBECOM 2002* (2002).

[10] CHERITON, D., AND GRITTER, M. TRIAD: A new next generation Internet

[4] See the archives of the NANOG mailing list, `http://www.nanog.org/`, for examples.

architecture, March 2000. Available from http://www-dsg.stanford.edu/triad/triad.ps.gz.

[11] CLARK, D. The design philosophy of the DARPA internet protocols. *Computer Communication Review 18*, 4 (Aug. 1988), 106–114. Proceedings of ACM SIGCOMM 1988.

[12] FELDMEIER, D., MCAULEY, A., SMITH, J., BAKIN, D., MARCUS, W., AND RALEIGH, T. Protocol Boosters. *IEEE Journal on Selected Areas in Communication (JSAC) 16*, 3 (Apr. 1998), 437–444. Special Issue on Protocol Architectures for 21st Century.

[13] FRANCIS, P., AND GUMMADI, R. IPNL: A NAT-Extended Internet Architecture. In *Proceedings of ACM SIGCOMM, San Diego, CA* (August 2001).

[14] GRITTER, M., AND CHERITON, D. R. An Architecture for Content Routing Support in the Internet. In *Proceedings of 3rd USENIX Symposium on Internet Techologies and Systems (USITS'01)* (March 2001), pp. 37–48.

[15] MORTIER, R., ISAACS, R., AND FRASER, K. Switchlets and resource-assured MPLS networks. Tech. Rep. UCAM-CL-TR-510, University of Cambridge, Computer Laboratory, May 2000.

[16] NAKAO, A., PETERSON, L., AND BAVIER, A. Constructing End-to-End Paths for Playing Media Objects. In *2001 IEEE Open Architectures and Network Programming Proceedings* (Ancorage, AK USA, Apr. 2001), pp. 117–128.

[17] NG, T. S. E., STOICA, I., AND ZHANG, H. A Waypoint Service Approach to Connect Heterogeneous Internet Address Spaces. In *USENIX Annual Technical Conference, Boston, MA* (June 2001).

[18] RENESSE, R. V., AND BIRMAN, K. P. Protocol composition in horus. Tech. Rep. TR95-1505, 29, 1995.

[19] SALTZER, J. H. Naming and Binding of Objects. In *Lecture Notes in Computer Science, 60: Operating Systems – An Advanced Course*, R. Bayer, R. M. Graham, and G. Seegmueller, Eds. Springer-Verlag, 1978, pp. 99–208.

[20] SRISURESH, P., AND EGEVANG, K. Traditional IP Network Address Translator (Traditional NAT). RFC 3022, IETF, Jan. 2001.

[21] STOICA, I., ADKINS, D., ZHUANG, S., SHENKER, S., AND SURANA, S. Internet indirection infrastructure. In *Proceedings of ACM SIGCOMM 2002, Pittsburgh, PA* (August 2002).

[22] TURANYI, Z., VALKO, A., AND CAMPBELL, A. 4+4: An Architecture for Evolving the Internet Address Space Back Toward Transparency, 2003. submitted to ACM Computer Communicatinons Review (CCR).

[23] VAN DER LINDEN, R. The ANSA Naming Model. Tech. Rep. APM.1003.01, Architecture Projects Management (APM) Limited, May 1993.

[24] VIXIE, P., ED., THOMSON, S., REKHTER, Y., AND BOUND, J. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136, IETF, Apr. 1997.

[25] WROCLAWSKI, J. T. The Metanet. In *Proceedings of the Workshop on Research Directions for the Next Generation Internet* (1997). Whitepaper, available at http://www.cra.org/Policy/NGI/papers/wroklawWP.