ISI-TR-2006-626

# A Recursive Network Architecture

Joseph D. Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90266 USA
+1 (310) 448-9151
touch@isi.edu

Yu-Shun Wang
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90266 USA
+1 (310) 448-8742
yushunwa@isi.edu

Venkata Pingali
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90266 USA
+1 (310) 448-8222
pingali@isi.edu

Oct. 20, 2006

## ABSTRACT[1]

The Recursive Network Architecture (RNA) explores the relationship of layering to protocol and network architecture. RNA examines the implications of using a single, tunable protocol for different layers of the protocol stack, reusing basic protocol operations across different protocol layers to avoid reimplementation. Its primary goal is to encourage cleaner cross-layer interaction and to support dynamic service composition, and to gain an understanding of how layering affects architecture. This document provides a preliminary description of RNA, its rationale, and discusses its features and challenges.

## 1. INTRODUCTION

The Recursive Network Architecture (RNA) reuses a single, flexible protocol for different layers of the protocol stack. RNA allows basic protocol operations to be reused in different protocol layers, avoiding recapitulation of implementation as well as encouraging cleaner cross-layer interaction. It allows protocols and protocol stacks to adjust at runtime, which allows more dynamic composition of services, both within stacks and in the way networking combines the stacks of individual hops into an overall network architecture.

RNA helps us explore the impact of layering on network architecture and avoid redesign of basic protocol constructs used in a variety of protocol layers. By providing a basic metaprotocol – a single protocol to be instantiated at all layers of a stack – RNA facilitates the composition of as-needed stacks at runtime, based only on the capabilities required over the regions desired. This extends the notion of a single configurable protocol, as in XTP and TP++, to retain the layering necessary to partition capabilities across regions (links, subnets, nets, ASes) in a network [7][10]. The resulting architecture makes it easier to apply a wide range of capabilities throughout the stack, to combine these layers dynamically, and to integrate related capabilities like security and congestion control – at different layers using a similar API.

## 2. Reasons for a new architecture

The current Internet architecture has been accused of ossification, but it has demonstrated numerous extensions over the years [19]. Various layers and capabilities have been added, including shim layers like SHIM6, HIP, security with IPsec and IKE, and TLS [8][12][13][16][18]. Some facilities have been added in new protocols, e.g., multiplexing in the SCTP transport layer and BEEP session layer protocols, and addressing in IPv6 [21][26].

Many of these extensions challenge the static nature of the conventional protocol stack, introducing alternate layers that must be selected at runtime. This particularly plagues the choice between IPv4 and IPv6, which is typically solved at the application layer rather than in the interface between transport and network. It also affects the choice between TCP, DCCP, and SCTP, which suffice equally well for a number of applications [14][20][26].

Adding new services to protocol layers often recapitulates services available at existing layers. TCP included state establishment and coordination from its inception in 1981; the need for similar 'connection' services have crept into a number of other layers, including tunnel protocols (MPLS, GRE), key exchange and filtering protocols (GRE, IPsec/IKE), sessions (BEEP), and other transport protocols (SCTP, DCCP, RTP) [9][12][13][21] [22][25].

Virtualization has been added at a variety of layers, including link (L2VPN), network (L3VPN, X-Bone, RON, Detour), and application (DHTs) [1][2][6][24][28]. Virtual layers, like shim layers noted earlier, add layers to a formerly static protocol stack.

[1] This material is based upon work supported by the National Science Foundation under Grant No. <CNS-0626788>. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The distinction between virtual and real layers is a somewhat artificial one, however.

RNA addresses these shortcomings of the current Internet architecture by providing a single, flexible architecture based on the reuse of a metaprotocol over different regions, and thus at different layers in the protocol stack. RNA reuses component services, such as three-way handshake, soft-state management, feedback-based congestion control, virtualization, and authentication at many protocol layers. It unifies the basic properties of a variety of protocols and protocol layers, and supports runtime protocol layer selection, enabling new dynamic stacks.

## 3. The Recursive Network Architecture

RNA is based on the notion that protocol stacks have design gaps, both between the layers (vertically), and between stacks (horizontally, also hop-by-hop), as shown in Figure 1. These gaps stem from the lack of understanding of how one protocol can link to or stack upon another (vertical), and how the forwarding operation (horizontal) integrates with traversing layers in a stack (vertical).

The *interlayer* (left) gaps affect next-layer resolution, where upper layer protocols are typically bound tightly to lower layer protocols, e.g., TCP being bound to IPv4 or IPv6 by the socket layer above TCP. The *interstack* gaps (right) affect next-hop resolution, in which each stack is typically bound to a particular network forwarding mechanism. RNA addresses these gaps, to enable protocol layers to be more coordinated within a stack and between different stacks throughout a network architecture.

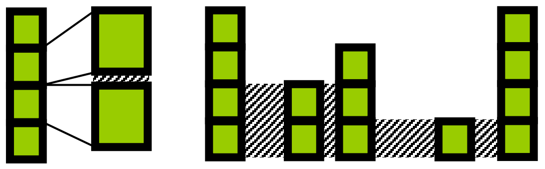


**Figure 1 Gaps in one stack (L) or between (R)**

RNA uses a single metaprotocol as a generic protocol layer. This metaprotocol includes a number of basic services, as well as hooks to configurable capabilities (Figure 2). This distinguishes RNA from configurable protocol systems such as Click and Netgraph; RNA develops the generic protocol, rather than just the software system in which it could be built. The metaprotocol includes interlayer coordination, such as might integrate identity management at various layers, or couple flow control.

This metaprotocol provides the building block from which protocol layers are formed. The architecture based on this metaprotocol is based on three fundamental observations about protocol design: that services are relative to a layer, that recapitulation (including virtualization support) should be avoided, and that composition (esp. dynamic composition) should be supported.

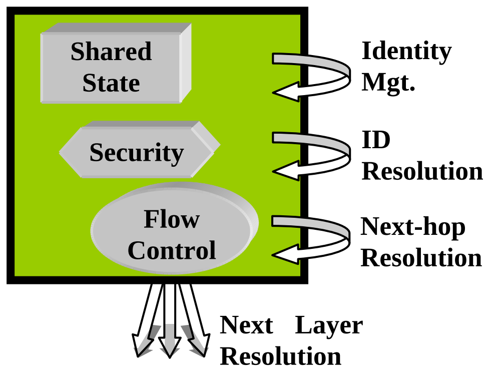


**Figure 2 RNA metaprotocol**

### 3.1 Observation 1: Services are Relative

There are a variety of services which can be implemented at multiple layers in a conventional protocol stack, including security, reliability, state management, policy (filtering), and congestion control. A number of earlier protocol systems were developed to try to modularize these capabilities, so that a particular instance of a protocol could have only the most efficient subset enabled. The eXpress Transfer Protocol (XTP) and TP++ were two such protocol systems. These systems assumed that only a key subset of desired capabilities should be enabled at any given layer [7][10].

It is useful to note that all services are relative, local only to the layer in which they are presented. Link security operates only over a single link hop; network layer security can protect the network layer, but is not sufficient for application layer security. This hints that we should revisit some aspects of the (in)famous End-to-End (E2E) Argument, which is based on the principle of non-composition [23]:

- **E2E Principle:** End-to-end services cannot be provided solely by the composition of hop-by-hop services**.**

As important as the principle is, there is a corollary which is often overlooked [27]:

- **E2E Corollary:** Hop-by-hop services may help performance, but they enhance, rather than replace, corresponding end-to-end services**.**

Networkers are familiar with the ISO 7-layer stack, in which each layer is imbued with a particular function, and provides particular capabilities (Figure 3). For example, the data link layer is responsible for formatting the data onto the next-lower layer (physical), and the network layer is responsible for multiplexing messages from the next-higher layer (transport). RNA notes that many of these services (including the two examples) occur at many layers in the stack; data formatting is also done at the presentation layer; multiplexing is also done at the session layer.

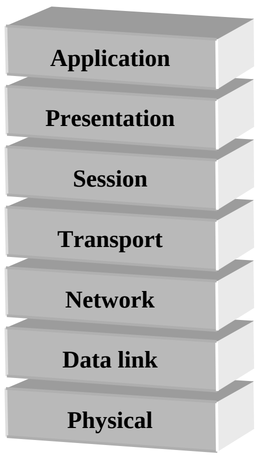


**Figure 3 ISO 7-layer reference model**

RNA observes that hop-by-hop services are the definition of layer-based services; all services within a layer are local to the endpoints of that layer. The E2E Corollary suggests that it might be useful to provide almost any service at a particular layer, notably in enhancing the performance of other layers.

The notion of a protocol layer is more than just a header format and processing rules. A layer exists relative to the layers it is between in the protocol stack. A layer is also local to a region, providing services only over those regions.

A layer builds on the services provided by the layers below, and provides them to layers above. Ethernet delivers frames between Ethernet endpoints, and IP delivers packets to IP endpoints – assuming a link layer such as Ethernet coupled with a forwarding mechanism at intermediate locations. Layers are also specific to regions; IP encompasses the public Internet, but a VPN encompasses that, plus private regions as well. IP is local to a pair of end systems, whereas HTTP is local to a pair of end applications.

RNA observes that the particular services of a protocol are context dependent, relative to the layers below and above, and that the services are local to its endpoints (Figure 4). There is little other difference between protocols, however. Protocols at the link, network, transport, and session layers may all require shared state to manage authentication and its associated filtering, but the distinctions between WEP, IPsec/IKE, TCP/MD5, and TLS are less significant. There are places where a particular protocol mechanism is better suited to its context – i.e., where stateless or soft-state coordination is better than hard state, but that is based on context more than layer per se.

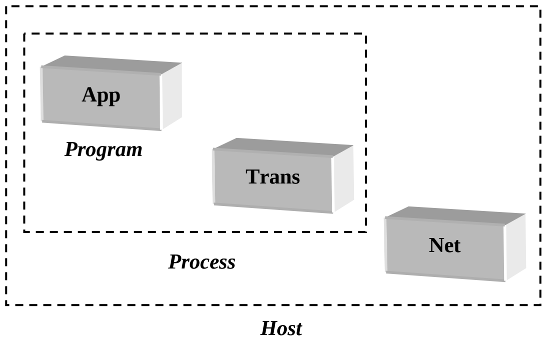


**Figure 4 Services are local to endpoints of a layer**

As a result of the current fixed layer architecture, new services are added either by wedging new layers between existing ones or by adding layers of virtualization (Figure 5, left and right respectively). Neither fits well within the current, static notion of a stack, and each begs the question of what services need to be added to existing protocol layers, and whether a new protocol is required to do so.

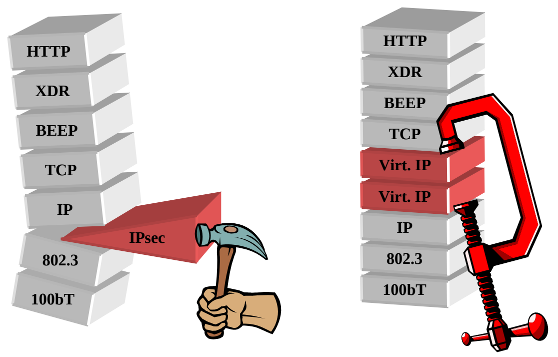


**Figure 5 Add shim layers (L) or virtual layers (R)**

As a result, in RNA, no service is specific to a particular layer; the same protocol, with a variety of services, suffices at any layer. That metaprotocol is tuned (manually or automatically) to the context at which it is deployed, and the same service may – and should – be deployed at a number of layers in the protocol stack.

## 3.2 Observation 2: Recapitulation is Avoidable

Offering similar services at many protocol layers invites recapitulation. There are many such examples, e.g., of security being re-implemented at the link, network, transport, session, and application layers, sometimes based on the same algorithms (e.g., DES or MD5). Some of these services include:

- handshake / state management
- security
- policy (admission control, filtering)
- multiplexing and demultiplexing
- retransmission
- reordering
- pacing / congestion control
- switching / forwarding

It is not always useful to implement all of these services at every layer of a protocol stack; reordering every layer could be very inefficient and unnecessary. It may be useful to reorder packets coming into a transport protocol to streamline processing, especially where processing predicts the structure of headers in sequence. Such prediction has been used to streamline TCP, and can be useful when the security algorithm involves chaining.

There is fundamentally little difference between the layers of an overlay vs. the base layers of the protocol that the stack natively supports. Virtualization layers need additional state coordination, to establish and maintain tunnels; they also require a switching/forwarding mechanism so that more than two tunnels can be joined at a node. A virtual protocol further requires a way to distinguish messages of one overlay from those of another; this involves labeling and a way to associate labels with context.

RNA observes that these capabilities are all part of any protocol at any layer, which is why it unifies them into a single metaprotocol. Additional layers of the metaprotocol can be invoked where desired, e.g., to enable particular services over subregions of a network, or to add services which have not been enabled by others at lower layers. This allows RNA's stack to accommodate both shim layers and virtualization layers without additional, separate mechanisms.

## 3.3 Observation 3: Composition Requires Coordination

The advent of component services at multiple layers requires coordination. Capabilities such as filtering, identity authentication, and congestion control require explicit coordination between layers. Connection latching and connection binding are examples of this.

Connection binding allows authentication from a higher layer to be used at a lower layer. This permits TLS identities, as might be derived from credit card information on web transactions, to be bound to IPsec associations [8][12][34]. Without network-layer protection, a TCP connection can be attacked and shut-down remotely, e.g., by sending RST messages [29]. Protecting the transport layer requires either transport or network layer authentication, so that only the valid endpoints of a connection can send packets affecting the signaling of the connection. However, the network layer security may not be configured with a single user's identity, e.g., on a shared host.

Connection latching is closely related to connection binding; latching ensures that data at one layer of a protocol is exchanged over a connection at that layer only when another layer is also connected [35]. This ensures, that traffic between a customer and website can proceed – at all layers of the stack – only when the appropriate layer indicates permission, e.g., when the user's credit card has been authenticated.

It is currently very difficult to combine connection binding and connection latching with existing protocols. TCP has one model of connection establishment, IPsec another (via IKE), and TLS yet another (at the application layer). Because these protocols use different mechanisms for state establishment and filtering, it is not always possible to merge their capabilities in a meaningful way without awkward external mechanisms. I.e., it is very difficult to establish an IPsec association via IKE with another endpoint that affects only a single TCP connection because IPsec filters on IP address and port numbers, but not on SYN cookies. There is no way to ensure that a particular TCP establishment (via a particular SYN) uses a particular IPsec Security Association (SA).

Further, DOS protection requires congestion detection and avoidance at all layers of the stack in order to reduce the effects of an attack. Every layer needs to be able to detect an overload, either of state, computation, or network resources, and to be able to shed that load, often at the previous layer. However, because each layer uses its own mechanisms for security, policy filtering, and load monitoring, it is difficult to integrate these mechanisms.

RNA observes that a single metaprotocol can support this coordination much more easily, since it uses a single API. The metaprotocol supports DOS offloading, filtering, and connection associations, so these features can be applied at any or all layers in the stack.

## 3.4 Features of RNA

RNA relies on the principle of a basic metaprotocol as a building block from which to instantiate layers and stacks. Each layer of the stack is an instance of the metaprotocol, tuned to the properties of the layers below it. Figure 6 shows four metaprotocol layers above a physical layer (shown unchanged here); in this case, mp-1 might employ retransmission if the physical layer were wireless, and might not if it were SONET. As a result, mp-2 might employ its own level of retransmission if any of the intermediate mp-1 hops had retransmission, to further limit the end-to-end retransmission delays. The layers of the metaprotocol are thus context dependent, but based on the same protocol; using the same mp-1 over a different physical layer results in a different mp-1' (Figure 7).

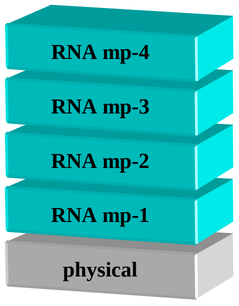


**Figure 6 RNA metaprotocol stack**

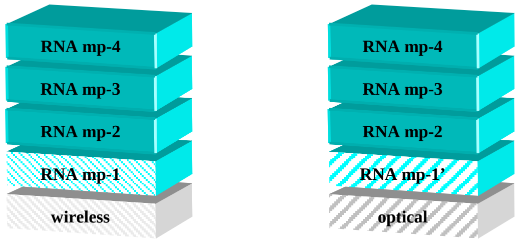
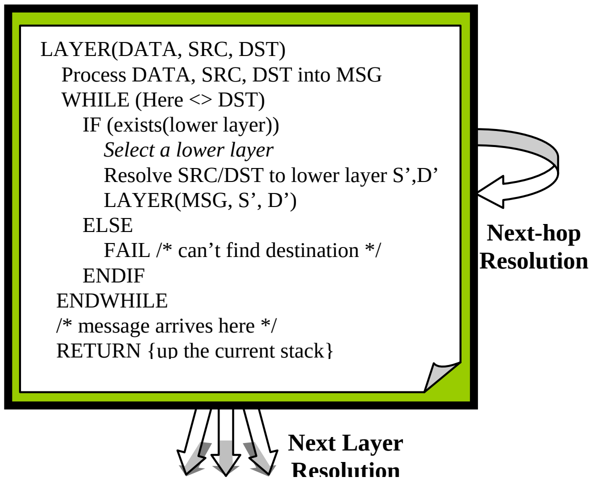


**Figure 7 RNA metaprotocol context-sensitivity**

This metaprotocol, shown in Figure 8, is based on a template developed for the X-Bone's Virtual Internet Architecture called the MultiDomain Communications Model (MDCM) (Figure 8) [33].



```
LAYER(DATA, SRC, DST)
  Process DATA, SRC, DST into MSG
  WHILE (Here <> DST)
    IF (exists(lower layer))
      Select a lower layer
      Resolve SRC/DST to lower layer S',D'
      LAYER(MSG, S', D')
    ELSE
      FAIL /* can't find destination */
    ENDIF
  ENDWHILE
  /* message arrives here */
  RETURN {up the current stack}
```

**Figure 8 MDCM template**

MDCM was developed to explain both next-hop and next-layer resolution, where either can be chosen at run-time. When a packet arrives at a protocol layer in a node, the steps shown in Figure 8 are followed. A packet is either at its destination address, or some other protocol – the next layer down, or the next hop – will hopefully get it closer to that destination. The line in italics, "*Select a lower layer*", is where both of these resolutions occur.

Note that MDCM is based on a structured, abstract template; packets are always formatted based on their source/destination addresses, protocol layers and hops are examined in turn, and the process continues at the next hop or next layer via a form of recursion (shown as "LAYER(MSG, S', D')"). Some of the services – such as 'Select a lower layer' (from among those extant) or 'resolve SRC/DST into lower layer S',D' are instantiated by plug-in functions. Configured various ways, MDCM instantiates the layer/hop behaviors of ARP, IP forwarding, DNS, and BGP.

RNA's metaprotocol is based on this kind of structure, additionally including some of the following steps. Note that the specific order, organization, and structure of the metaprotocol is part of the RNA research, but it will include:

1. Establish / refresh state
2. Encrypt / decrypt message
3. Apply filtering
4. Proceed based on flow control
5. On input, proceed based on reordering
6. Multiplex/demultiplex as indicated (includes switching/forwarding)

These are the protocol operations which, in the MDCM template, are denoted by “Process DATA”, and otherwise ignored in that model. RNA thus extends MDCM to address not only resolution functions between the layers and stacks, but also the other mechanisms common to all protocols at all layers.

RNA takes advantage of this metaprotocol to encourage reuse of common mechanisms, including hard (handshake-based) and soft (refresh-based) state maintenance, security, filtering, etc. This unified approach makes it easier to integrate related functions at different layers, such as coupled authentication (i.e., Channel Binding), coupled connections and filtering (i.e., Connection Latching), and coordinated reordering, retransmission, and flow control.

## 4. Challenges

RNA explores the impact of the structure of a network stack and protocols therein on network architecture. There are a number of open issues in RNA which make it exciting, exploratory research. These include the design of the metaprotocol itself, the system that manages the stack, ways to integrate interaction between layers in the stack, and the extent to which context and performance plays a role in how the metaprotocol operates.

### 4.1 Metaprotocol Design

RNA relies on a single metaprotocol to provide the capabilities currently dispersed in different layers of the protocol stack. The composition and sequence of the component services are key to developing a flexible module that can be used in a variety of contexts.

The services will be chosen from among the list provided in Section 3.2, where some well-known examples include:

- state management: hard state from TCP; soft state from RSVP
- congestion control: feedback-based from TCP and its extensions
- security association: key exchange from IKE

Due to the limited scope of the project, simplified versions of these protocols will be incorporated or code reused from existing implementations where possible. In addition, the sequencing and priorities of the component services will reflect current best-practices for protocol design, e.g., to filter early, to act only on authenticated fields (when authentication is available), etc.

### 4.2 Stack Management

Part of deploying the same metaprotocol at different layers, as well as over different regions of a network assumes a stack management system. Individual messages need to be labeled with the chain of applicable metaprotocol instances, and nodes need to be able to install and remove metaprotocol instances. For this purpose, existing modular protocol processing systems, such as Click or Netgraph will be employed [15][17]. Coordination between stacks will be done either by using a predefined list of protocol chains – as is implicitly the case for existing, ‘well-known’ protocols in the Internet (e.g., DNS, SNMP, DHCP), or by using a negotiation phase that an instance of the metaprotocol can itself provide.

### 4.3 Interlayer Coordination

A major challenge for existing protocol stacks is the coordination of services between protocol layers, such as would support Connection Latching and Channel Binding [34][35]. Connections initiated at one layer often depend on services at other layers, including filtering, authentication, and state establishment. Although RNA’s reuse of a single metaprotocol module facilitates this coordination, the APIs the module provides for such interaction are critical to this capability. Important aspects of these APIs are the order, locking, and nesting of transactions across the layers, which is another key area of research for RNA.

### 4.4 Context and Performance Sensitivity

It would be impractical to expect every instance of the RNA metaprotocol to operate identically. Different instances need to be tuned to the context in which they are deployed, i.e., to use soft state, hard state, or no shared state as desired. This tuning is expected to be part of protocol deployment, i.e., when new physical layers are added, or additional processing capabilities available at a node and protocol layer, the adjacent layers would adjust accordingly.

It would be useful for the metaprotocol to ultimately self-tune, to adapt as needed, enabling or disabling services on-the-fly based on the capabilities of adjacent layers. By disabling services not needed, a layer could increase its performance. Different layers could also coordinate their services, e.g., bidding for use of a fixed (application-specified) latency budget, where reordering could be deployed where it is most useful, subject to an overall reordering latency impact.

## 5. Related work

RNA is related to previous work on modular protocol systems, though it advocates for the use of a single instance – the metaprotocol – rather than just supporting a template where protocols can be instantiated. Other, template-based protocol systems have also been considered, but tend to focus on a single protocol layer, whereas RNA relies on the use of the metaprotocol at different layers in a stack to operate over different regions of a network, just as existing stacks do. RNA attempts to incorporate the kind of context-sensitivity that other research has shown useful in deploying existing Internet protocols in nonstandard environments. Finally, RNA is inspired by configurable, integrated protocols, but again intends a solution that can be used recursively throughout the protocol stack rather than at a single layer.

## 5.1 Modular Protocol Systems

The Click modular router, x-Kernel, and Netgraph systems all support modular design of network protocols. Click focuses on router functions, and operates on a variety of platforms [15]. The x-Kernel focuses on the integration of modular messaging into the operating system [11]. Netgraph is a more incremental approach, supporting dynamically loadable and inter-connectable packet processing functions as a configurable kernel service [17]. RNA differs from these services in that our metaprotocol is an instance of a protocol, not a template from which others might develop a protocol. RNA will use one of these services – likely Netgraph or Click – as the platform for developing both the metaprotocol module and the dynamic stack management system.

Flexible Protocol Stacks similarly provides an environment in which protocols can be implemented, providing a rich set of services from which an instance can be developed [32]. RNA similarly relies on a set of common services; we will examine leveraging their work in this area.

## 5.2 Template-based Protocol Models

RNA's metaprotocol is based on an extension of the X-Bone's MultiDomain Communication Model (MDCM) [28][30][33]. MDCM focused on the interaction between the layers of protocols in a stack and on the impact of next-hop and ID resolution on the dynamic structure of a stack, but ignores the basic operation of a protocol. RNA completes the MDCM design, adding the rest of the services of a protocol, such as state management, security, filtering, etc., rendering MDCM's template into a functional protocol instance.

USC/ISI's Role-Based Architecture (RBA) proposes a heap-style header in a single-layer architecture, in which individual protocol modules ('roles') can be implemented in a variety of ways and addressed explicitly [5]. RBA removes the concept of layering, and replaces it with a rule-based association between Role-specific headers, allowing them to be reordered as desired. RNA relies on the conventional concept of layering, but uses the replication of a single protocol throughout different layers in different network regions. RNA could incorporate RBA's more flexible header structure and processing sequence in its metaprotocol, and might also be able to order its layers in ways that map to roles, but this is orthogonal to the goals of the RNA project.

## 5.3 Context-Sensitive Extensions

A number of protocols have incorporated context-based extensions for enhanced performance. The generic concept of adding such features, typically at intermediate nodes using shim protocol layers, are known as Performance Enhancing Proxies (PEPs) [4]. PEPs can be deployed over subsets of a network to introduce limited retransmission or deploy a tunnel with forward error correction, such as to correct increased error over mobile wireless networks. Similar extensions have been developed for existing protocols, such as wireless-aware TCP [3].

RNA integrates these capabilities within the layers of the protocol stack, so these capabilities can be deployed as needed less obtrusively. In particular, RNA allows TCP-like retransmission to be supported over a link layer because the metaprotocol supports retransmission that can be enabled as needed. This avoids the need to create intermediate overlay layers, such as PEPs, or even some overlay management systems (Detour, RON, X-Bone, DynaBone) support [1][24][28][31]. It also avoids the need to redefine host-to-host semantics (i.e., that loss probably means congestion) in TCP to accommodate the characteristics of only a portion of a network (i.e., the wireless mobile part).

## 5.4 Configurable Protocols

RNA is based on the concept of the generic, inclusive protocol, such as the eXpress Transfer Protocol (XTP) and TP++ [7][10]. These protocols included a large set of services, such that enabling or disabling particular services could instantiate different protocols, like connectionless messaging, transaction messaging, and reliable streams. Both protocols considered the current view that services should be specific to a particular layer of a protocol stack, and intended to be configured to provide only those services that were necessary at their transport layer.

The Stream Control Transport Protocol (SCTP) is a more recent protocol designed in a similar spirit [26]. SCTP was originally developed to support transaction-based messaging services, in particular as would be needed to support telephone signaling (SS7 in particular) over the Internet. It included a number of features considered necessary for telephony, including support for backup addresses for fault tolerance, which is evolving into support for multipath communication. SCTP borrows heavily from TCP's connection management and congestion control protocols, and adds a layer of multiplexing that allows a single connection to support multiple, concurrent transfers.

RNA is designed to apply the concepts of these generic, configurable protocols recursively throughout a networking stack. By including numerous services at every protocol layer as an option, it is hoped that RNA will encourage the use of these services at the layers where they are most useful. This distinguishes RNA from SCTP, where all such services are supported within the transport layer alone. Unlike XTP and TP++, RNA assumes that services can be reused at multiple layers in the stack and thus operate over different regions of a network; this avoids the need to determine which layer is solely responsible for a particular service. Whether to enable it at any particular layer can be a comparatively local (subject to performance and interaction) decision.

## 6. Summary and Future Work

This paper presented an overview of RNA, the Recursive Network Architecture. RNA applies a single protocol across different layers and regions, to explore the relationship between layer environment and scope on network architecture. This work is just commencing; the next step involves a preliminary implementation to determine how protocol reuse at multiple layers impacts our notion of layer capability and differentiation.

## 7. REFERENCES

[1] Andersen, D., Balakrishnan, H., Kaashoek, M., Morris, R., “Resilient Overlay Networks,” Proc. 18th ACM SOSP, Banff, Canada, Oct. 2001.

[2] Andersson, L., and Rosen, E., Editors, “Framework for Layer 2 Virtual Private Networks (L2VPNs),” (work in progress), June, 2004.

[3] Balakrishnan, H., Seshan, S., Amir, E., and Katz, R., “Improving TCP/IP Performance over Wireless Networks,” ACM Conference on Mobile Computing and Networks, Oakland, CA, November 1995.

[4] Border, I., Kojo. M., Griner, J., Montenegro, G., and Shelby Z., “Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations,” RFC 3135, June 2001.

[5] Braden, R., Faber, T., and Handley, M., “From Protocol Stack to Protocol Heap - Role-Based Architecture,” Proceedings of the First Workshop on Hot Topics in Networking (Hotnets-I), ACM SIGCOMM, Princeton, NJ., October 2002.

[6] Callon, R., and Suzuki, M., “A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs),” RFC 4110, July, 2005.

[7] Chesson, G., Eich, B., Schryver, V., Cherenson, A, and Whaley, A., “XTP Protocol Definition Revision 3.0,” Tech. Rept. Protocol Engines, Inc., 1900 State Street, Suite D, Santa Barbara, CA 93101, Jan., 1988.

[8] Dierks, T., and Allen, C., “The TLS Protocol Version 1.0,” RFC 2246, January 1999.

[9] Farinacci, D., Li, T., Hanks, S., Meyer, D., and Traina, P., “Generic Routing Encapsulation (GRE),” RFC 2784, March 2000.

[10] Feldmeier, D.C. “An Overview of the TP++ Transport Protocol,” in: Tantawy A.N.(ed.): High Performance Communication, Kluwer Academic Publishers, 1994.

[11] Hutchinson, N., and Peterson, L., "The X-Kernel: An Architecture for Implementing Network Protocols," IEEE Trans. on Software Eng., 17(1), pp. 64-76, 1991.

[12] Kaufman, C., (ed.), “Internet Key Exchange (IKEv2) Protocol,” RFC 4306, December 2005.

[13] Kent, S., and Seo, K., “Security Architecture for the Internet Protocol,” RFC 4301, December 2005.

[14] Kohler, E., Handley, M., Floyd, S. and Padhye, J., “Datagram Congestion Control Protocol (DCCP),” Internet Draft draft-ietf-dccp-spec-02 (work in progress), May 2003.

[15] Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M., “The Click Modular Router,” ACM Transactions on Computer Systems, 18(3), pp. 263-297, 2000.

[16] Moskowitz, R., and Nikander, P., “Host Identity Protocol Architecture,” (work in progress), August 2005.

[17] "Netgraph: Graph-Based Kernel Network Subsystem," FreeBSD manual page, http://www.freebsd.org.

[18] Nordmark, E. and Bagnulo, M., "Level 3 Multihoming Shim Protocol," draft-ietf-shim6-proto-03.txt (work in progress), September, 2005.

[19] Peterson, L., Shenker, S., Turner, J., "Overcoming the Internet Impasse Through Virtualization," Proc. Hotnets-III, Nov. 2004.

[20] Postel, J., "Transmission Control Protocol," RFC 793, September 1981.

[21] Rose, M., "The Blocks Extensible Exchange Protocol Core," RFC 3080, March 2001.

[22] Rosen, E., Viswanathan, A., and Callon, R., "Multiprotocol Label Switching Architecture," RFC 3031, January 2001.

[23] Saltzer, J., Reed, D., and Clark, D., "End-to-end arguments in system design," ACM Transactions on Computer Systems 2, 4 (November 1984) pages 277-288.

[24] Savage, S., Anderson, T., et al., "Detour: a Case for Informed Internet Routing and Transport," IEEE Micro, V19, N1, Jan. 1999, pp. 50-59.

[25] Schulzrinne, H., Casner, S. Frederick, R. and Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications." RFC 3550 / STD 64, July 2003.

[26] Stewart, R. and Xie, Q. "Stream Control Transmission Protocol (SCTP): A Reference Guide," Addison Wesley, New York, NY, 2001.

[27] Touch, J., "End2End Argument vs. Programming the Internet: Are the Two Complementary?" presentation at E2E issues panel at Opensig 2000, Napa, CA, Oct. 2000.

[28] Touch, J., "Dynamic Internet Overlay Deployment and Management Using the X-Bone," Computer Networks, July 2001, pp. 117-135. A previous version appeared in Proc. ICNP 2000, pp. 59-68.

[29] Touch, J., "Defending TCP Against Spoofing Attacks," (work in progress), Feb. 2006.

[30] Touch, J., Wang, Y., Eggert, L. and Finn, G., "Virtual Internet Architecture," presented at Future Developments of Network Architectures (FDNA) at Sigcomm, August 2003. Available as ISI-TR-2003-570.

[31] Touch, J., Finn, G., Wang, Y., Eggert, L., "DynaBone: Dynamic Defense Using Multi-layer Internet Overlays," Proc. 3rd DARPA Information Survivability Conference and Exposition (DISCEX-III), Washington, DC, USA, April 22-24, 2003, Vol. 2, pp. 271-276.

[32] Tschudin, C., "Flexible Protocol Stacks," Proc. ACM SIGCOMM 1991, pp. 197-204, 1991.

[33] Wang, Y, Touch, J., Silvester, J., "A Unified Model for End Point Resolution and Domain Conversion for Multi-Hop, Multi-Layer Communication," ISI Tech. Report ISI-TR-2004-590, June 2004.

[34] Williams, N., "Clarifications and Extensions to the GSS-API for the Use of Channel Bindings," draft-ietf-kitten-gssapi-channel-bindings-01 (work in progress), Oct. 2005.

[35] Williams, N., "IPsec Channels: Connection Latching," (work in progress), Feb. 2006.