Recursive Temporal Namespaces

Venkata K. Pingali
USC Information Sciences Institute
4676 Admiralty Way,
Marina del Rey, CA 90292
+1-310-448-8222
pingali@isi.edu

Joseph D. Touch
USC Information Sciences Institute
4676 Admiralty Way,
Marina del Rey, CA 90292
+1-310-448-9151
touch@isi.edu

ABSTRACT
Recursive temporal (RT) namespaces extend spatial namespaces such as IP with the notion of a nested time interval. RT namespace is an architectural approach to dealing with the uncertainty associated with changes to distribution, syntax or semantics of large-scale namespaces that are associated with high cost and probability of failure. We discuss the abstract design issues associated with RT namespaces and two existing approaches that are special cases of the general design. We also propose a variant of IP based on RT namespaces that supports open-ended evolution.

1. INTRODUCTION
Names, such as IP addresses, are used to identify the endpoints of communication. Namespaces see two kinds of changes; we observe both of these in the case of IP. The first is commonly called renumbering or renaming, and involves the redistribution of names. The second corresponds to the shift from IPv4 to IPv6. This kind of change is relatively infrequent, and involves modifications to syntax or semantics of the namespace. Both these are traditionally handled as one-off events, and are associated with significant unbounded costs of disrupting existing communication and services. Mechanisms that reduce the risk associated changes reduce the barrier to change and make the network more evolvable [7].

The Recursive Temporal (RT) namespace solution eliminates the notion of timeless or absolutely persistent namespace. Instead, a set of namespaces is used, in which some namespaces are more persistent than others. By overlapping the lifetimes of the namespaces, the network supports changing or evolving in an endless fashion. We organize these namespaces to make the process more tractable. The overall expressiveness of the network also increases through temporal name resolution and routing.

ReArch'08, Dec 9-12, 2008, Madrid, Spain.

This paper proposes the use of RT namespaces to enable open-ended evolution in namespaces. RT namespaces extend spatial namespaces such as IP with a notion of lifetime. This simple idea is developed into a recursive structure with a well defined set of operations. We discuss the design issues associated with RT namespaces, and show that existing proposals such as Shadow configurations [1] and space-time contexts [8] are instances of RT namespaces. We propose Evolvable IP, an instance of an RT namespace that is deployable today and supports the open-ended evolution of IP.

2. WHY ADD TIME TO NAMESPACES?
Changes in protocol name definition and distribution occur for many reasons, including efficiency, scale, functionality and administration. In most cases, this is not an issue because the changes tend to be small in scale or infrequent, or because communication disruption is acceptable. However in some cases, none of these reasons are true, and change is difficult. The fundamental problem underlying change is the uncertainty associated with the goals, process, and outcome of change. Small changes are known to have high impact on the network [6]. The change from IPv4 to IPv6 has proven to be much harder than initially thought because of technical and economic issues. The nature of uncertainty in each case is different, and so are the approaches to deal with them. Uncertainty is important because real world services depend on the correctness and availability of the namespace and communication, and any disruption extends beyond the namespaces into the real world. Ubiquitous long-lived namespaces such as IP support a large number of real world services. The impact of unbounded disruption can range from simple inconvenience to damage to life and property. In the common case, network disruption lead to disruption of valuable services such as business communication [6] and health services [2].

Consider the simple case of two-node communication shown in the Figure 1. Node X discovers the existence of node Y and path L to Y, and stores the result of the discovery in memory. When Y is renamed, X's memory is updated. The restoration of communication state is followed by the restoration of application state. Syntactic and semantic changes to the X-Y communication can be

modeled as changes in the computation at both ends. The change is instantaneous, *i.e.*, there is an instant of time when the old communication state ceases to exist and the new state comes into effect. This change model has the value that it is simple, but it assumes complete information about the nature and process of change. The question is primarily one of efficiency of the change process.

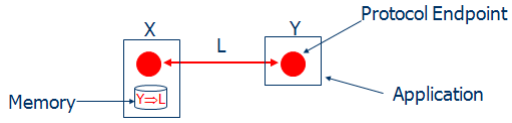

Figure 1 Simple model of communication

This change model is not appropriate for all changes in large-scale long-lived namespaces. First, we may not know about the correctness of the changes, *i.e.*, whether the changes are meaningful and useful. Second, the locations where the discovered names are stored in memory. such as inside firewalls and applications, may not be known. Third, the number of nodes that are impacted by a given change may be large. In such cases, a large amount of coordination may be required at the instant of change, and experience suggests that it is hard to impose strict time bounds on distributed consensus algorithms with imperfect processes. Fourth, we may not know how the disruption of propagates in the entire network due to other nodes' dependency on X-Y communication. Last, the impact of the disruption on the application and therefore the real world may be unknown.

This requires us to find a much more incremental path to change that enable dealing with the uncertainty involved in the process of change. However, at every point in time the modified namespace must be integrated with the rest of the system to minimize disruption and enable testing.

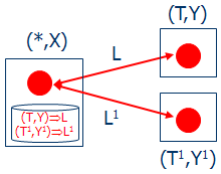

Figure 2 Namespace extended with time

Time is introduced into namespaces as shown in Figure 2. We associate a lifetime or time interval with each name, and store the temporal information associated with the name along with spatial information. When the node Y is renamed to Y^1, the state of Y is not modified; instead, a copy of the state is created that is named Y^1. While the new relationship $X\text{-}Y^1$ is being established, the X-Y communication can continue uninterrupted. This may not be possible in the unmodified system due to correctness issues arising from concurrent use of conflicting names. Once $X\text{-}Y^1$ is tested and found to be correct, communication using X-Y path is migrated to $X\text{-}Y^1$. Introducing time decouples the time of change from the time when the change is committed or used. This simple idea is developed further below to understand the full implication of introduction of time into namespaces. The resulting capabilities include recursion, mapping and transformations.

3. RECURSIVE TEMPORAL NAMESPACE

The previous section introduced the basic notion of temporal names as a two-tuple (time interval, spatial name). Our notion of time interval is relative, and our focus is on the relationships between time-intervals rather than absolute values. Examples of relationships between time intervals include inclusion, in which one time interval covers another time interval, sequencing, in which one interval occurs after another interval, and concurrency, in which two intervals may overlap. The representation of time itself may be explicit or implicit. The time intervals can be thought of as belonging to a namespace that supports partial order among the names. Eventually this namespace will be renamed or refined. That, in turn, leads another temporal namespace in order to support changes to this namespace, *ad infinitum*, limited by the computation, memory and management abilities of the network.

A temporal namespace is a partial ordered set of names, and a temporal name is a member of that set. A spatial namespace is a protocol-defined namespace in which names are assigned to distinct protocol endpoints based on some spatial attribute, such as abstract, topological or geographical location. RT namespaces are effectively directed acyclic graphs (DAGs) with spatial names as the leaves and various temporal names as non-leaf nodes. DAG edges indicate the temporal inclusion relationship between names. Figure 3 shows a sample organization of RT namespaces with t_i representing the temporal, and s_j representing the spatial names. In the next section, we discuss two existing examples of RT namespaces.

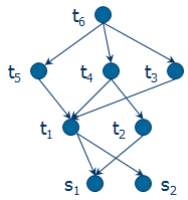

Figure 3 One possible organization of namespaces

A RT name is a path $(t_{i1},t_{i2},...,t_{ik},sj)$ in the DAGs from the root to the leaf. Groups of RT names that differ in one node name may be represented using a single path that uses a group or set name instead of atomic node names. Each host in the network has as many RT names as there are paths from the leaves to the root – all of which may be simultaneously valid and usable. In practice there might be multiple such DAGs at any one time each having a different change dynamic, distribution, security or other properties. The organization shown may be the view from a host in one part of the network, and the view may be

different from another part, if the spatial scopes are not identical, which we expect to be the common case.

Spatial namespaces are created due to differences in functionality, administrative boundaries, trust, performance *etc*. Temporal namespaces are created to deal with changes of various kinds and over different timeframes. Both are transformed over a period of time. Together they determine the structure of the DAG and choices for other design parameters. The organization shown in Figure 3 is somewhat arbitrary, and discovering meaningful organizations is a key challenge. In the next section, we discuss two recent proposals that are single-level. The proposed Evolvable IP, discussed later, has two levels.

4. EXAMPLES

In this section we consider two recent research proposals from the recursive temporal naming perspective. They both incorporate single-level time into their system design.

4.1 Shadow Configurations

Shadow configuration [1] is a mechanism to deal with reconfiguration of routers in networks. IPv4 routing and forwarding functions are extended to support two concurrent configurations, called *real* and *shadow*. The *real* configuration is actively used to forward packets, and the *shadow* is the new configuration that is being prepared. The *real* configuration is used to set up the *shadow* and is eventually replaced by *shadow*, at which point *shadow* is renamed *real* and the existing *real* becomes the *shadow*. The namespaces are isolated from each other and there is no forwarding between the two configurations. Internal data structures of routing processes are extended with temporal naming. The configuration and testing applications were modified to incorporate the notion of *shadow*. The isolation on the wire is achieved using two bits in the IPv4 header. The deployment of *shadow* is coordinated across the nodes using a custom distributed algorithm. The authors also present significant amount of work on other complementary mechanisms that allow performance measurement of the shadow configuration and switching between the two.

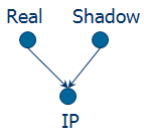

Figure 4 Shadow configurations

Shadow configurations can be modeled as a single-level RT namespace with temporal namespaces with two members, *shadow* and *real*, as shown in Figure 4. Both namespaces have identical spatial scope. The RT namespaces discussion suggests that if applications such as VoIP and VPNs are deployed in the network, then the correctness of the new configuration requires more extensive testing and the switching process is more complicated. In such cases, mapping of names and forwarding between *real* and *shadow* may be required.

4.2 Space-Time Contexts

Space-time contexts [8], upon which the idea of RT namespaces is based, associates each name with a region of interpretation called a context. Figure 5 shows a simple deployment of three contexts over three nodes. The objective of contexts is to enable renaming or redistribution of names with minimal disruption. The use of contexts is similar to that of Shadow configurations. A new context is deployed when a part of the network is renumbered, and the old context is deleted. An arbitrary number of contexts, each with a different scope, may be deployed. Contexts are dynamically deployed and integrated using cross-context forwarding. The implementation extends IPv6 with a new inter-context routing header within packets, inter-context forwarding tables at specific nodes, separate and simple spatial and temporal routing, and user-level processes to coordinate the decentralized construction of the contexts.

Space-time contexts can be modeled as a specific kind of temporal namespace that is single-level and flat, with varying spatial scope, and limited temporal forwarding. The RT namespaces discussion suggests that temporal mapping between names would be eventually required, and the single-level time assumption would have to be dropped.

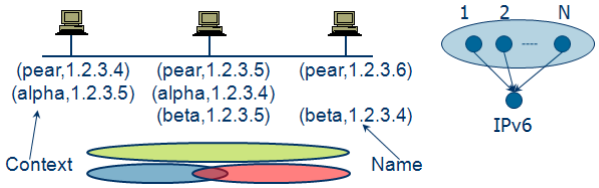

Figure 5 Three contexts deployed over a small network

5. DESIGN ISSUES

Introducing time into namespaces raises a number of issues and increases operational complexity. The fundamental value of temporal extension is the separation of the *time of change* from the *time of use*. This moves uncertainty to a place where it can be handled better, at the price of additional complexity. This tradeoff has also been observed in economics and systems engineering. Detailed knowledge of the deployment scenario is necessary to determine the complete solution and tradeoffs. We discuss some of the high-level design issues in this section.

5.1 Host Architecture

The host architecture must be modified to support isolation between the states associated with various names. Interfaces to application and higher protocol layers must be extended to support binding to an arbitrary subset of names. Routing requires intra-host discovery of RT names, and forwarding requires delivery of messages across RT namespaces. Dynamic deployment of temporal namespaces

requires intra-host cross-namespace capabilities covering state management and testing. The changes are significant, but the value gained through reduced disruption and smoother evolution is significantly higher.

5.2 Protocols and Messages

At an abstract level, the boundaries of the protocol become fuzzy as changes to syntax and semantics are incorporated over time. It is more appropriate to talk about an aggregate protocol, or a set of protocols instead of one. The design of such a protocol set is incomplete, by definition. The key change in the protocol state and message structure is in terms of the notion of protocol endpoint. The source and destinations are expressed as path in the RT namespace instead of an atomic spatial name. These paths change over the long term. Further, multiple names may be used to exploit multiple-paths in time in addition to space. This capability enables separation between control and data paths of protocols in space-time instead of space-only.

5.3 Forwarding and Routing

The complexity of routing increases in four ways. First, there is multi-level temporal routing in addition to spatial routing. The two might be separate or integrated depending on the deployment scenario. Efficient route computation may or may not be possible depending on the degree of opaqueness and compatibility of spatial routing. Second, integration and removal of temporal namespaces from the routing system results in convergence issues and inefficiency. Third, expressiveness of the routing systems must be increased to support avoid specific routes for testing or other purposes. Last, the introduction of time divides the Internet into timeful and timeless regions. One simplifying factor is that the deployment is not expected to cross enterprise boundaries, and therefore eliminates some of the problems that make BGP-like protocols complex. Further, constraints can be imposed over the frequency of creation of namespaces, deployment scope or syntax/semantics to reduce the space of possible routing algorithms and enable simpler implementations.

Although routing may be complex, forwarding may be relative simpler. Forwarding at the 'leaf nodes' is spatial-only and timeless. Each leaf node has a unique spatial forwarding table appropriate for the syntax and semantics of that namespace. Forwarding at every other level is temporal. A message that cannot be forwarded at a given level tries to forward the message at the next higher level. If no such level can be found, then the message is dropped. Temporal forwarding can happen only between names that are co-located and spatial forwarding between names that are at one-hop distance or less. The structure of the forwarding tables depends on the design of the RT namespace. The examples herein impose additional constraints to make the forwarding and routing manageable.

5.4 Coordination

Every step during the lifetime of an RT namespace involves coordination across hosts, including creation of the namespaces, discovery, switching, and garbage collection. The nature of coordination required depends on the deployment scenario. The examples discussed in this paper include a custom distributed algorithm in case of Shadow configurations, and decentralized mechanism in case of space-time contexts.

5.5 Decision Problems

The complexity of a system grows quadratically or worse with the number of temporal namespaces introduced, because the number of logical protocol endpoints increases linearly, but the number of possible pairs of communication increases quadratically. A number of new decision problems are introduced that deal with the timing and nature of namespace transformation and use, and parameters such as the spatial scope. Early results [8] show that computing the cost of disruption and the parameters of the temporal namespace are hard (NP-Complete or harder) even under strong assumptions. Therefore deployments of RT namespaces are not expected unless the value proposition is clear. The examples discussed in this paper focus on scenarios including mobility, upgrades, and reconfiguration.

6. NAMESPACE TRANSFORMATION

Change happens over time through transformation of the RT namespace, *i.e.*, modifying the DAG. A number of operations are possible on the basic DAG that add or delete nodes and edges. The modifications add and delete state, and sometimes add and delete software. We discuss each of these in this section.

6.1 Expand

Expansion is the process of adding nodes to the DAG. Expansion is typically executed when the current organization of namespaces is inadequate for capturing changes to distribution, syntax or semantics of the namespace, new nodes are added to enable deployment of the modifications.

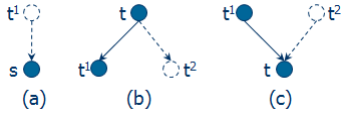

Figure 6 Operations for building recursive namespace

Figure 6 shows three operations that are used to build the namespace, each with a different purpose. Figure 6(a) shows addition of a parent to an existing node. This corresponds to a "fork-lift" upgrade of the namespace from being a purely spatial entity to becoming a space-time entity. This is expensive because this involves significant host software changes and changes to the way network is managed. Figure 6(b) shows a child t^2 being added to an

existing node t that already has a child t^1. The addition of the edge indicates that the time interval represented by t includes the time interval represented by t^2. There may or may not be temporal forwarding between t^1 and t^2. Figure 6(c) shows the case where a new temporal t^2 is added that has the spatial scope and substructure of t^1. The two are 'siblings', and may support forwarding. This is typically used for renumbering. Both shadow configurations and space-time contexts support dynamic addition of temporal names as shown in Figure 6(c), but they are at only a single level. They differ in terms of the timing and lifetime of the expansion, and nature of the nodes added.

6.2 Collapse

Collapse is the process of deleting nodes and edges in the DAG. Figure 7 shows the three basic collapse operations on the recursive temporal namespace. There is one operation corresponding to each operation shown in Figure 6. A name, and possibly the corresponding namespace, is deleted when the change process is completed. All communication is moved to other names/namespaces and the specific name/namespace represented by the nodes is deleted. In practical terms, this involves clean up of data structures corresponding to the names. Figure 7(a) and Figure 7(b) shows a name t^1 being deleted. This primarily involves cleaning up the state associated with t^1. Figure 7(c) shows the case of a single parent that is being deleted, *i.e.*, there are no other members of the temporal namespace to which t^1 belongs. In this case, in addition to the cleanup of state, the software associated with the temporal namespace may be deleted as well. Both shadow configurations and space-time contexts support deletion of nodes as shown in Figure 7(b). In case of shadow configurations, there is only one node to remove, but multiple nodes may be removed simultaneously in space-time contexts.

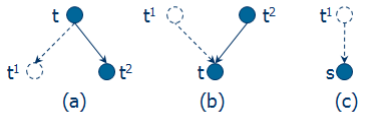

Figure 7 Operations for collapsing recursive namespace

6.3 Map

Given the flux in an RT namespace, there is a need to store temporal mapping between names. This is useful for migration and integration purposes. Existing mapping services such as DNS may be extended or some other mechanism may be introduced. The structure of the map depends on the implementation, and may vary from a centralized to a distributed entity. Figure 8 shows an abstract representation of two maps between RT namespaces. Neither shadow configurations nor space-time contexts support mapping. IPNL [4] is an example of a simple single-level recursive temporal namespace that uses temporal mapping between names. The mapping is primarily intended to support host mobility but can be extended to support renaming.

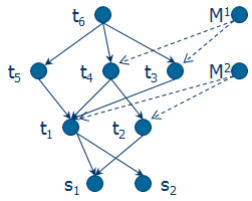

Figure 8 Temporal mapping services

7. Evolvable IP

In this section we propose a protocol, Evolvable Internet Protocol (EIP), based on IPv6. The objective of EIP is to incorporate the notion of open-ended evolution – both short term as well as long term – into the protocol itself assuming that the fundamental notions of packets and unicast destinations does not change. It is an instance of a two-level RT namespace, *i.e.*, has two levels of time. The first level is aimed at short-term reconfiguration, and the second level is aimed at longer-term changes in syntax and semantics. EIP is effectively a collection of protocols in which each is replaced by another over an appropriate timeframe.

Figure 9 shows an organization of contexts intended to enable transition from IPvX that could very well be IPv6 to a hypothetical IPvY and beyond. The first level temporal namespaces $\{t_{10}, t_{11}\}$,$\{t_{00}, t_{01}, t_{01}\}$ and $\{t_{20},t_{21}\}$ represent different address distributions of underlying spatial namespace such as IPvX. The second level names $\{t_0,t_1,t_2\}$ represent long-term changes to syntax and semantics of IP. Figure 10 shows the packet header structure in which the source and the destination are two-level RT names.

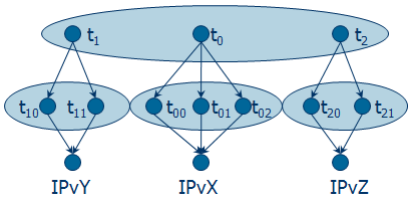

Figure 9 EIP scenario involving three versions of IP

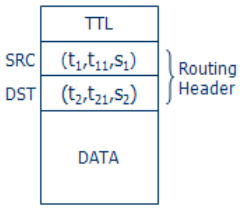

Figure 10 Example EIP header

A simple prototype implementation of EIP is under construction. The prototype builds on experience with space-time contexts. Contexts already support first-level

temporal names for IPv6. Our EIP implementation extends context implementation with a second-level temporal routing table, as well as other features. A simplified IPv6 message format and forwarding capability is used as the "base" on which to build EIP. Two aspects of IPv6 enable the implementation of EIP – the support for link-local addressing and the support for generic loosely-constrained routing headers. The use of link-local addressing effectively turns IPv6 into a link-layer protocol for EIP. EIP source and destination addresses are encoded in the routing header.

Support for contexts already involves a significant amount of change to the host network stack covering address assignment to interfaces, generic header processing in IPv6, other network-layer protocols such as ICMP, transport-layer protocols such as TCP, and the socket interface. Applications such as *route*, *ping* and *traceroute* were modified as well. EIP requires generalization of these changes, an additional forwarding table, and user-level commands. The EIP forwarding algorithm is relatively simple. At each node, the second level temporal name, *e.g.*, t_2, is looked up to determine the next hop at the same level, *e.g.*, t_0. If the latter is reachable within the host, then the message is delivered to the namespace. If not, a lookup is performed to determine the next hop at the next level, *e.g.*, t_{11}. The process is repeated again at the first level. There are additional issues associated with transport protocols, application interfaces, and address assignment to interfaces that are addressed in the prototype. Beyond forwarding and basic reachability across IP versions, EIP requires extensions along several fronts including routing, mapping, coordination and semantics.

This additional complexity is the price paid to manage risk associated with a given change to IP. On the positive side, multiple alternative future IP versions can be concurrently pursued as a way to deal with uncertainty in timing, correctness and value of individual changes.

8. RELATED WORK

Naming has been studied extensively [9]. Several aspects of naming systems deal with time, including the lifetime of the namespace itself, lifetime of the names and discovery [3], versioning or change over a period of time [11], and changes to syntax and semantics of the namespace. Our RT namespace adds yet another explicit but limited notion of lifetime to namespace that is dynamic, distributed, and recursive. A single-level RT namespace is equivalent to decentralized versioning on multiple trees. An RT namespace is similar to a shadow configuration [1] in that it is an application of virtualization [7][1] to the problem of change. RT namespaces complement existing network management approaches [5][10] by addressing a new class of high risk changes [2][6]. From an architectural point of view, the framing of locator/identifier split debate is altered by the support for change over time within locators themselves.

9. CONCLUSION AND FUTURE WORK

The paper presented the design issues associated with recursive temporal (RT) namespaces. Two existing research proposals were shown to be special cases of our more general design. A new proposal for Internet layer namespace virtualization based on RT namespaces, called Evolvable IP (EIP), was also presented. Future work includes the continued development and implementation of EIP, and demonstration of its feasibility. Other issues that are being investigated include routing, coordination, and switching processes.

10. ACKNOWLEDGEMENTS

This work was partly supported by the National Science Foundation (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 NSF.

11. REFERENCES

[1] Alimi, R. *et al.*, "Shadow Configuration as a Network Management Primitive," SIGCOMM, Aug 2008.

[2] Computer World, "The VA's computer systems meltdown: What happened and why," Nov 20, 2007.

[3] Droms, R. (Ed.), "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," RFC 3315, Jul 2003.

[4] Francis, P., Gummadi, R., "IPNL: A NAT-Extended Internet Architecture," SIGCOMM, Aug 2001.

[5] HP OpenView., Available at http://www.hp.com

[6] International Herald Tribune, "RIM traces BlackBerry outage to poorly tested software update," Apr 20 2007.

[7] Peterson, L. *et al*, "A Blueprint for Introducing Disruptive Technology into the Internet," HotNets-I, 2002.

[8] Pingali, V. K., "Addressing Uncertainty during Renaming using Space-Time Contexts," Ph.D Dissertation (in preparation), USC/ISI, 2008.

[9] Comer, D. E. *et al*, "Understanding naming in distributed Systems," Distributed Computing 3(2), Jun 1989.

[10] Stallings, W., *SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management*, Addison-Wesley Longman Publishing Co., Boston, MA, 1993

[11] Tichy, W. F., "RCS - A System for Version Control," Software Practice & Experience 15:7, July 1985.

[12] Touch, J. *et al.*, "Virtual Internet Architecture," ISI Tech. Rep. ISI-TR-2003-570, Mar 2003.