

# Addressing Uncertainty during Renaming using Space-time Contexts

Venkata K. Pingali  
USC Information Sciences Institute,  
Marina del Rey, CA, USA  
touch@isi.edu

Joseph D. Touch  
USC Information Sciences Institute  
Marina del Rey, CA, USA  
pingali@isi.edu

**Abstract**— Protocol names such as IP address change over time, and such changes result in disruption of communication. This disruption is acceptable in most circumstances, but not all. In case of latter, additional approaches are required to deal with the fundamental problem associated with the disruption – uncertainty of scope, nature and timing of the impact. This paper suggests an architectural mechanism, called space-time contexts, to deal with this uncertainty. Space-time contexts allow conflicting distributions of names to co-exist, which enables an alternative model of change. The paper also describes the design issues associated with contexts and our current implementation.

**Keywords**—Architecture; Network Protocols; Network management; Internet;

### III. OVERVIEW

Names, such as IP addresses, are used to identify endpoints of communication. Renaming is the process of altering the spatial distribution of protocol names in layered protocol networks. Renaming is a significant part of a larger problem of (re)configuration of networks, and is expensive in terms of human coordination effort and high risk in terms of the potential for disruption of communication [6][9]. However, renaming is inevitable. Networks merge, split or alter connectivity due to business decisions, and changes in hardware and software implementations. The fundamental problem associated with renaming is uncertainty - not being able to anticipate the scope and cost of the disruption caused by renaming. This uncertainty arises from incomplete knowledge about the specific change and information relationships that are affected by the change, incomplete control over the network, and imperfect implementations of endpoints and protocols. This problem becomes significant as the number of nodes in the network increases.

We suggest an architectural approach to reduce risk of disruption during renaming. Names lack a built-in notion of time, *i.e.*, names have no past, present, or future. By extending namespaces such as IP with limited notions of time, which we call *space-time contexts*, change can be handled as a first-class operation in the network. Space-

time contexts enable an alternative model of change by altering the nature of information that is discovered and used for communication. Beyond change, this architectural mechanism also enables other advanced capabilities such as automatic configuration and protocol markets.

The rest of this paper presents further discussion of the renaming problem, a description of the approach and the design issues that arise, and a prototype that is under development.

### IV. UNCERTAINTY DURING RENAMING

Names must be discovered by protocol nodes in a network in order to communicate, *i.e.*, construct message headers and determine the availability and route to the destination. An NFS client (Figure 1), for example, must discover the IP address or domain name of an NFS server before it can communicate. The number of individual protocols such as NFS and the number of deployments for each is increasing every day. An enterprise network typically has tens of thousands of nodes and dozens of active protocols.

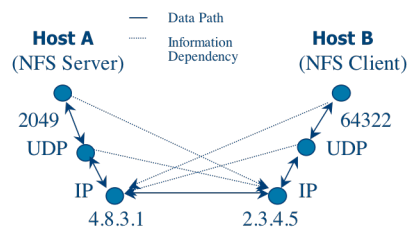


Figure 1. End-to-end path with name dependencies

The discovery of protocol names expensive in terms of computation and time, and often involves a variety of non-trivial single and multi-hop protocols such as SNMP and DNS, and out-of-band mechanisms involving humans. The discovered names are stored in the network at endpoints in configuration files, directories such as DHCP and DNS, and control nodes such as firewalls, often in an uncoordinated way. These stored names create information dependencies across protocol nodes.

Renaming protocol nodes has two basic consequences. The ongoing communication is disrupted and information dependencies are invalidated. The communication failure may trigger other failures and propagate through the system. There is uncertainty associated both these aspects.

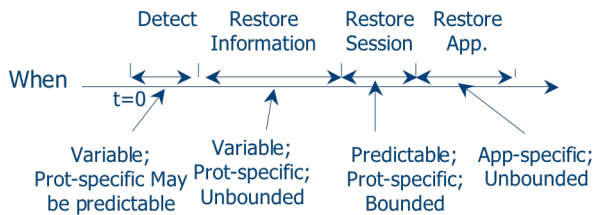


Figure 2. Various stages in the renaming process

Figure 2 shows four distinct stages of the renaming process: detect, restore information dependencies, restore session, and restore application. Communication can proceed only when this process is complete and correct. Further, the process is sensitive to where the change occurs and who is initiating the communication. In the example shown in Figure 1, it does not typically matter if an NFS client is renamed, but the impact is much higher if the server is renamed instead, *e.g.*

The first stage of renaming involves detecting the change. The time to detect varies with the protocol and may be bounded in some cases. Connection-oriented protocols, such as TCP, terminate in a predictable way. Connectionless protocols require either the application or some other mechanism detect that the endpoints have changed. As a result, the change may not be noticed for a long time, or not at all.

The second stage involves rediscovering the information required to re-establish the communication. The structure of information dependencies determines the endpoints where discovery must be performed. The discovery process is on the critical path to communication restoration, and any uncertainty associated with discovery has a direct impact on the duration of the disruption. In a large enterprise network, there is higher uncertainty due to lack of coordination across discovery processes and fragmentation of knowledge.

The third stage involves restoring protocol session state. This is dependent on the individual protocol. This stage is typically predictable and short due to well-characterized protocol state machines, predictable network latencies, and endpoint availability.

The fourth stage involves restoration of non-protocol state of the applications. Applications rely on a composition of protocol endpoints. They tend to have complex state machines and the impact of the disruption

on this state is not always predictable, due to explicit and implicit assumptions about the system, environment, and use. The impact may extend beyond the disruption of communication into the real world, *e.g.*, if the application is used to perform essential control or service function. Recovery from these errors may be more expensive than the cost of restoration of the communication itself.

Disruption of the application and forced rediscovery are the reasons for why renaming is more than the act of changing the name and being able to reach the modified names, *i.e.*, routing convergence. Reachability is necessary but not sufficient for the correct completion of the renaming process. Uncertainty is in general increasing over time due to increased scale of deployment, diversity of the protocols, and complexity of the applications. Complementing these is the current economics of network management, which encourages short-term decisions and staff turnover, leading to loss of tacit knowledge about the network. The cost of disruption is also increasing over time due to growth in economic activity based on the Internet. At the same time the required pace of changes is increasing due to newer devices, usage patterns and market conditions. All these contribute to the problem of change becoming more challenging over time and also more frequent.

### III. SPACE-TIME CONTEXTS

A *space-time context* is a region of interpretation for protocol names such as IP addresses. The context-extended names are represented as a tuple (C,n) where n is the name and C is the region within the name is valid. Although the syntax and semantics of name n is specified by the protocol, the context name C could have protocol-independent structure such as a string, a version number or a sequence number. Each context has a spatial scope and a lifetime. Note that we are referring to a version of spatial distribution of names, and not protocol version itself, such as versions 4 and 6 for IP.

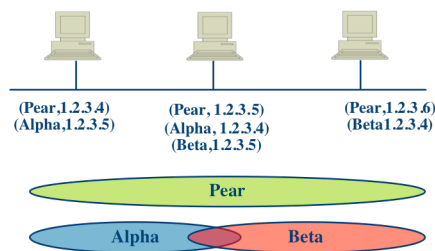


Figure 3. Three contexts with conflicting name distributions

Consider the example network configuration shown in Figure 3. The space-time name (Pear, 1.2.3.4) identifies “Pear” as the region within which the IP address 1.2.3.4 is meaningful. The word “Pear” is meaningful to the human operators but has no semantics associated with it

within the system. The context isolates this IP address from all other IP addresses in other contexts. Beta, for example, also has the same IP address but assigned to a different node in the network.

Contexts add to complexity and resource usage. A context is additional information that is represented and manipulated throughout the system. Applications and configuration files are extended to support contexts. Applications bind to IP addresses from one or more active contexts available at a given node, and specify the destination context in addition to destination IP address during communication. Existence of an IP address in one context does not affect IP address or ongoing communication in another context. This is because the end-to-end communication from the sender application to the receiver application is isolated along the entire path. IP packets sent between hosts include the context information so that the packet can be correctly processed at each hop. Inter-context forwarding tables must be constructed at each node to enable message delivery across context boundaries.

The figure shows contexts at a given point in time. The set of contexts and the spatial scope of the contexts may change over time. Each node must therefore allow dynamic creation and management of contexts and the IP addresses within. The creation of a new context is coordinated across multiple nodes. The coordination process cannot assume that there is a ‘base’ context that is persistent, covers all nodes and available all the time. All nodes may not support contexts and/or may have syntactically or semantically incompatible notions of contexts. Additional mechanisms may be necessary to deal with such cases, which further add to the complexity of the system.

#### IV. RENAMING WITH CONTEXTS

The process of renaming using can be divided into six distinct steps - 1. Compute 2. Deploy 3. Discover 4. Test 5. Select 6. Integrate. Each of these steps is a design space that has its own set of challenges. Each space allows multiple approaches depending on the feasibility and cost of the approach used. The relative importance of these steps will vary with each usage scenario. A detailed discussion of the design spaces is available in related work [10].

The process is renaming is shown in Figure 4. Let the current context be called “Pear”. Pear represents a particular distribution of IP addresses and IP-related state at various locations including access control rules, firewall and routing. The change process starts with the computation of the new context “Gamma.” The new objectives are reflected in the distribution of names within Gamma along with the nodes at which it is

deployed. Deployment of Gamma may coincide with deployment of other contexts from layers above and changes to the software. Gamma is tested for correctness, performance and other characteristics of interest. Pear context provides the communication paths required for coordination between all nodes involved. Once Gamma is deployed and tested, the process integrating the new context into the existing set of contexts begins. This involves first adding Gamma to the inter-context routing, and second, coordinating the process of switching higher-layers from Pear to Gamma.

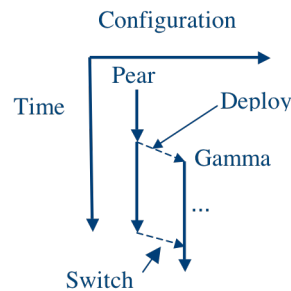


Figure 4. Timing relationship between Pear and Gamma

We make several observations about this process here. There may be disruption during the process of creating the new context, Gamma. The degree of the disruption depends, among others, on the details of isolation mechanisms provided within the host and on the link. Testing of the new context overlaps with ongoing communication in Pear. Gamma may very well turn out to be faulty and useless in which case it could be discarded. The existence of contexts lowers the correctness requirements of the computation that determines the details of the change. There may be potential disruption during the process of integration of Gamma into the inter-context routing system. However that disruption is a function of the routing protocol and may be easier to limit than discovery process. The creation of Gamma is not immediately followed by use. The applications still use the old context Pear. The decoupling of the name creation/change from the time of its use is critical to addressing the problem of instantaneous impact of change in today’s network.

#### V. DESIGN ISSUES

There are a number of design issues related to contexts as an architectural feature including scalability issues arising from increase in quantity of information, the nature of relationship between contexts, host architecture required to support contexts, the nature of inter-context routing and forwarding, the coordination required to automatically deploy the contexts, and methods used to determine the boundary of the context, Depending on the nature, number, and distribution of the

contexts being deployed, the relative importance of each of the issues varies. A more complete and detailed discussion of the issues may be found elsewhere [10].

**Scaling.** A key problem associated with space-time names is the explosion in the space of possible paths between nodes. The effective number of nodes in the network increases by  $k$ , where  $k$  is the average number of active contexts at each node. The number of paths, and therefore the computational and memory cost of routing, increases quadratically. A second problem is that representation of multiple names for each layer in the message headers is likely to be prohibitive. Although it is possible for each node have names from arbitrary and unrelated contexts, it is expected that the context creation and assignment across nodes and layers satisfy some small number of patterns. These patterns create redundancy in time and space that can be used to make the problem of managing the context information more tractable.

**Context organization.** The notions of time that are captured in the space-time names and their relationships include (1) relative time, (2) inclusion, and (3) transitivity. The notion of time suggested here is relative and not clock based. The space-time scope of one context may be included within another space-time context. This notion is binary, *i.e.*, one context is included in another or not. We do not quantify the inclusion in a relative or absolute sense. The inclusion is transitive, *i.e.*, if context A includes context B, and B includes context C, then A includes context C. The set of contexts is partially ordered and can be represented as a directed acyclic graph.

**Host architecture.** Context require new capabilities within the host to support: (1) isolation between context-specific states (2) ability to discover other contexts on a given host and dynamically create/destroy them, (3) ability to group contexts from multiple layers to construct multi-layer context-stacks, and (4) ability to associate context-state from across hosts. This is provided using extended control architecture for the protocols. There are three basic components in the architecture: modules, namespace and interfaces. Together they allow (1) dynamic instantiation of contexts, (2) discovery and manipulation of intra-host contexts, and (3) control and data message delivery across contexts.

**Routing.** Inter-context communication arises due to the fact there is interesting computation and data stored in different contexts. The reasons differ in space and time, and include servicing of clients, backward compatibility, and state migration. Forwarding across space-time contexts requires a combination of information specified in the message header, *viz.*, one or more space-time names of the destination, and information stored in

forwarding tables to traverse the various contexts. This is similar to inter-domain routing in the Internet. Some of the issues in inter-context forwarding include (1) the nature of information contained in message headers, (2) mechanism to deliver messages across contexts, (3) the inter-context forwarding table structure and content, and (4) inter-context routing protocol.

**Coordination.** The deployment and management of contexts requires coordination across multiple hosts. The need for architecture for coordination arises from the fact that there is no one strategy that is appropriate for all times and scenarios. Coordination is required at multiple steps during the process of renaming including discovery, deployment, garbage collection, testing and switching. The function performed in each is different and there is value in reusing the same basic coordination capability for all. We can look at each coordinated operation as a coordinated invocation of specific primitives at each of the nodes and processing of the results of the invocation. Together, they address the following five questions: what primitives to invoke, where to invoke them, the parameters/arguments for the primitives, the specific algorithm to coordinate the invocation, and how to process the result of the operation. The exact answer for each depends on the details of the implementation.

**Boundary determination.** As noted earlier, there is both a gain and a cost associated with each context. The gain is in terms of reduced risk of disruption during change. There is increased gain as the size of the context increases. At the same time there is increased switching and resource cost. Thus, there is a tension between the two. Determining the boundaries of the context is a key challenge. Our current approach, not discussed here, uses simple risk analysis to understand the tradeoff. Our early results show that this decision problem is NP-Complete [10].

## VI. RELATED WORK

The questions of change and evolution of large system touch a number of different areas including software engineering, network management and network architecture. Software engineering is aimed at improving developer productivity and makes assumptions about knowledge, control and communication. Network management has focused on methods to configure and diagnose network protocols and devices but does not address the fundamental issues underlying the problem of uncertainty. Network architecture emphasized scale, semantics and simplicity but does not, in most cases, take the evolution of the system into consideration.

Some approaches of software engineering include [4] software repositories to store all components and versions, cross-platform compositional frameworks to

build and manage complex software systems, project management processes, formal architecture description languages and tools, long-term evolution [6]. The work closest to contexts is software versioning. However contexts and versioning differ along a number of dimensions including distribution of nodes and control, degree of connectivity, and cost structures.

Network Management is a broad term that is used to refer to the methods to configure and monitor networks. It involves protocols such as SNMP and CMIP[12], configuration databases, and methods. It is a rich problem space and approaches have been developed for various problem sub-spaces including models, vendor-specific tools, workflows, and social mechanisms. All the above are used in isolation and together in real world networks. From an architectural point of view, none of them alter the basic nature of renaming in layered networks. In many cases, they reduce the scope, cost and time of the change so that the difficult aspects such as uncertainty, recursion and layering effects can be dealt with humans administrators. Space-time contexts aims to alter the underlying change model for a subset of changes. It addresses recursion by creating control paths in space-time instead of space-only.

The notion of space-time context is similar to the notion of autonomous system in the Internet, the Region in Metanet [13], a Virtual Internet in the Virtual Internet Architecture [13], and naming context in both Plutarch [4] and FARA [2]. The differences between architectures themselves and with space-time contexts include the purpose of the construct, how instances of the construct are created and managed, how these instances are integrated, how the system scales with the number and distribution of instances, and degree of automation. Most of these previous works assumes that individual contexts occur in a single protocol layer and the relationship between contexts is primarily spatial. Further, the coordination required to deploy instances of the construct are assumed to be outside the system. Almost all of them do not consider change/renaming as a first class operation. Space-time contexts is a particular instantiation of the general notion of contexts underlying these architectural proposals that is intended to reduce disruption during particular class of changes.

## VII. PROTOTYPE

We implemented a prototype to test the feasibility of the ideas presented in this paper. The prototype uses a combination of FreeBSD 7.0 Beta 2 operating system [6] extended with Clonable stacks [15], IPv6 which we extended with a inter-context routing header and forwarding logic, VLANs to provide link-level isolation, and our user-level software to dynamically deploy

contexts. In addition to demonstrating the feasibility of contexts, the computational overhead associated with contexts was also measured at the router. The our implementation was found to be approximately 20% slower than an unoptimized kernel under moderate loads.

For simplicity, the contexts are assumed to be uniform, each with a name from a global 32-bit context identifier space. A new IPv6 inter-context routing header was added and FreeBSD extended to handle this new header. IPv6 forwarding was also extended to perform inter-context forwarding at the edge of each context. In order to achieve this, a new inter-context forwarding table was introduced that is looked up when IPv6 packets cannot be forwarded within the context anymore. Intra-host isolation between context-specific states was provided using Clonable Stacks extensions to the default kernel. This extension replicates the data structures associated with each protocol layer in the kernel such that they are all isolated from each other. The key advantage is that the socket interface is kept unmodified, and therefore applications run unmodified within each context. Each context is mapped to one stack within the host. The default clonable stack is extended with ability to discover other stacks, context identifier information, and inter-context forwarding. Instances of the stacks across hosts were composed into a context using VLANs. Other options that were experimented with include tunnels and hop-by-hop headers. TCP/IP code was extended to handle the contexts as well. Various components including the control block, hash table, checksum computation, and syncache were modified to support contexts. Socket interface was extended to support context-aware applications.

A simple system for deployment of contexts was designed that supports multiple deployment strategies. The deployment system consists of control processes running within each context that expose a set of context management primitives. These primitives are invoked in a coordinated fashion based a coordination program that specifies the location and timing of the invocations. The program is distributed to all nodes at runtime using a simple distributed consensus algorithm, Echo. The capabilities are minimal and will be extended in future.

The experimental setup for measuring forwarding overhead is shown in Figure 5. All the machines run the modified kernel described above. The machines at each end acted as traffic sources and the machine in the middle as the forwarder. The endpoint machines were 2.4GHz Intel P4 dual-core processors with 1GB RAM and PCI Intel Pro/1000 Gigabit Ethernet interface. The forwarder was configured with 2.4GHz Intel Core 2 Duo processor with two PCI Intel Pro/1000 Ethernet interfaces. All the network interfaces used the older 82540 chipset. The

machines were connected back-to-back to avoid any interference from cross traffic.

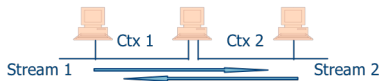


Figure 5. Experimental Setup – Three machines with two contexts and streams of packets

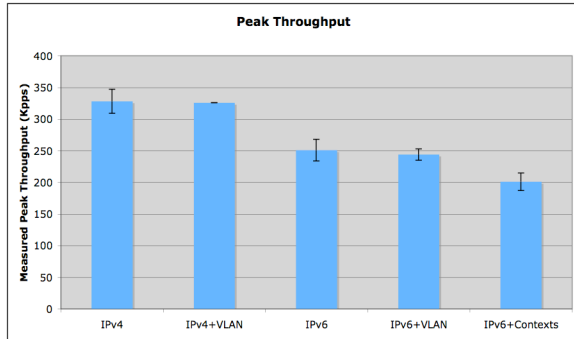


Figure 6. Peak throughput under moderate loads

Figure 6 shows the relative peak performance of inter-context forwarding under moderate loads. The labels have the following meanings: IPv4 = IPv4 forwarding across regular interfaces, IPv4 + VLAN = IPv4 forwarding across VLAN interfaces, IPv6 = IPv6 forwarding across regular interfaces, IPv6 + VLAN = IPv6 forwarding across VLAN interfaces, and IPv6 + Contexts = Forwarding of IPv6 messages with inter-context routing header across VLANs that are in separate network stacks. We make several observations here. IPv6 forwards at a slower rate than IPv4. The primary reason for this slowdown in this particular version of FreeBSD is that the IPv4 code is multi-threaded whereas IPv6 code is not. IPv4 code is better able to hide latency compared to IPv6. The cost of VLAN is minimal for both IPv4 and IPv6. Inter-context forwarding is about 20% slower than IPv6. This is primarily because of the additional computation involved. This path has further optimization opportunities including caching of the route entries, reducing unnecessary copies of the message, and reducing queuing.

### VIII. STATUS AND FUTURE WORK

The prototype is under active development along with a dependency graph-based model aimed at quantifying the tradeoff between the cost and benefit of using

contexts. Beyond this the notion of space-time contexts may be developed along at least a couple of different directions. First, we could look at sequence of changes instead of single change. This work really addresses the problem of unit change, *i.e.*, one change step in isolation. However, in order to achieve a given correctness, performance and other goals of the system, we may have to perform multiple changes in a particular sequence. Applications of the sequencing include protocol markets and automatic configuration. Second, we could develop the notion of contexts further through applications to multiple layers and recursively to contexts themselves. This could lead to a more general notion of contexts [10].

### REFERENCES

- [1] F. Baker, E. Lear, and R. Droms, "Procedures for renumbering an IPv6 network without a flag day," IETF RFC 4192, September 2005. <http://www.isi.edu/innotes/rfc4192.txt>
- [2] R. Braden, T. Faber, and M. Handley, "From protocol stack to protocol heap -- role-based architecture", ACM CCR, Vol. 33, No. 1, January 2003, pp. 17--22.
- [3] D. Clark, R. Braden, A. Falk, and V. Pingali, "FARA: Reorganizing the Addressing Architecture," in Proceedings of the ACM SIGCOMM FDNA Workshop, August 25-27, 2003, Karlsruhe, Germany.
- [4] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield, "Plutarch: an argument for network pluralism," In Proc. of the SIGCOMM FDNA Workshop, Aug. 2003, Karlsruhe, Germany.
- [5] J. Estublier, "software configuration management: a roadmap," in Proc. of the Conference on The Future of Software Engineering, June, 2000, Limerick, Ireland, pp 279-289.
- [6] The FreeBSD Project, <http://www.freebsd.org>
- [7] M. M. Lehman, "Approach to a theory of software evolution," Eighth International Workshop on Principles of Software Evolution (IWPSE), September 2005.
- [8] mValent, Inc., 2007 Market Survey Challenges and Priorities for Fortune 1000 Companies, 2007, <http://www.mvalent.com>
- [9] News.com, "Commentary: Microsoft outages hold universal lessons," <http://www.news.com/2009-1001-251651.html>
- [10] V. K. Pingali, "Space-time contexts: addressing uncertainty during renaming," Ph.D Dissertation, USC/ISI, in preparation.
- [11] M. Rochkind, "The source code control system," In IEEE Transactions on Software Engineering SE-1:4, December, 1975, pp. 364-370.
- [12] W. Stallings, SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management, Addison-Wesley Longman Publishing Co., Inc., 1993, Boston, MA,
- [13] J. Touch, Y. Wang, L. Eggert, and G. Finn, "Virtual Internet Architecture", ISI Tech. Report, ISI-TR-2003-570, March 2003.
- [14] J. T. Wroclawski, "The metanet," Workshop on Research Directions for the Next Generation Internet, May 1997.
- [15] M. Zec, "Implementing a clonable network stack in the FreeBSD kernel," In Proc. of USENIX Annual Technical Conference, FreeNIX Track, June 2003, San Antonio, TX.