Designing for Scale and Differentiation

Karen R. Sollins
MIT Laboratory for Computer Science
March 26, 2003
Submitted for publication

Abstract

Naïve pictures of the Internet frequently portray a small collection of hosts or LAN's connected by a "cloud" of connectivity. The truth is more complex. The IP-level structure of the Internet is composed from a large number of constituent networks, each of which differs in some or all of transmission technologies, routing protocols, administrative models, security policies, QoS capabilities, pricing mechanisms, and similar attributes. On top of this, a whole new structure of application-layer overlays and content distribution networks, equally diverse in the sorts of ways mentioned above, is rapidly evolving. Virtually any horizontal slice through the current Internet structure reveals a loosely coupled federation of separately defined, operated, and managed entities, interconnected to varying degrees, and often differing drastically in internal requirements and implementation. Intuitively, it is natural to think of each of these entities as existing in a region of the network, with each region having coherent internal technology and policies, and each region managing its interactions with other regions of the net according to some defined set of rules and policies.

*In this paper, we propose that a key design element in an architecture for extremely large scale, wide distribution and heterogeneous networks is a grouping and partitioning mechanism we call the **region**. Furthermore we postulate that such a mechanism can provide increased functionality and management of existing unresolved problems in current networks. The paper both describes a proposed definition of the **region** concept and explores the utility of such a mechanism through a series of examples. We claim that there is significant added benefit to generalizing the idea of the **region**.*

1. Introduction*

We have seen orders of magnitude growth in networks and the Internet. As elements on networks become smaller, their numbers will only increase further. Predictions of vast sensor nets, including "smart paint", "smart cement", "smart buildings" suggest magnitudes of numbers of elements increasing by many orders of magnitudes. In this paper, we propose that a key design element in an architecture for extremely large scale and wide distribution networks is a grouping and partitioning mechanism. Furthermore we postulate that such a mechanism can provide increased functionality and management of existing unresolved problems in current networks.

* This research was funded under NSF Grants 0122419 (ITR/SY: Center for Bits and Atoms) and 0137403 (Regions: A new architectural capability in networking) and a gift from the Cisco University Research Program.

Virtually any horizontal slice through the current Internet structure reveals a loosely coupled federation of separately defined, operated, and managed entities, interconnected to varying degrees, and often differing drastically in internal requirements and implementation. Intuitively, it is natural to think of each of these entities as existing in a region of the network, with each region having coherent internal technology and policies, and each region managing its interactions with other regions of the net according to some defined set of rules and policies.

This paper will explore the broad utility of a concept we call the *region*. A region is an entity that encapsulates and implements scoping, grouping, subdividing, and crossing boundaries of sets of entities. In network systems, these functions are used for a variety of purposes including scaling, heterogeneity, security, billing, performance, trust management, and so on. The assumption that we explore in this work is that we can separate mechanism from purpose, by providing a single highly optimized and reusable generic mechanism to serve a number of purposes. The original Internet architecture had no concept

of region. To meet a variety of needs the idea has been introduced in an ad hoc way in many places. At the core of this paper we will be exploring what might happen if it were introduced as an architectural capability, as originally suggested by Wroclawski [Wro97].

The region captures two basic concepts, the group and its boundary. The entities within a region share a set of characteristics or invariants; by being in a region, an entity is defined to reflect the invariants of that region. The key characteristic of the boundary is that we can know approaching or crossing a boundary may trigger an action, such as either modification of some piece of state or notification transmitted to an appropriate entity. In this work, a boundary is a logical concept, although we often describe the idea by analogy with topological or physical boundaries. Orthogonal to these basic ideas, the region will be capable of adaptive re-organization or optimization under changing conditions. This optimization must be managed under the constraint of a well-defined specification of the region interface. Because the behavior of the generic region abstraction will be well defined and reusable as well as optimizable, regions will be scalable and widely available.

We expect the generalization to provide two key improvements to the current situation. First, it will relieve the implementer of re-inventing mechanism and provide access to adaptably improved behavior. Second, a shared region abstraction will provide a number of benefits including both a new paradigm for managing the flow of information across layer boundaries, and the opportunity for mutual improvements between layers. This framework for multi-layer interaction will enhance rather than eliminate the layered model of networking.

This paper explores a particular definition or specification of a region and proposes it as a first class object in the network architecture. Section 2 defines regions in some detail. Section 3 discussions an initial set of issues in designing adaptive regions. Section 4 describes several example situations in which different aspects of the region concept would be valuable. Section 5 reviews related work, and Section 6 concludes with an abstract interpretation of regions and a summary of the contributions of this paper.

2. Defining a region

The introduction hinted at a definition of a region. Further definition is best done by discussing the types of operations or functions a region supports. These fall into three groups, definition, membership, and boundary management functions. We will discuss each group separately.

2.1. Creating and deleting a region

There is a small set of issues related to the creation and deletion regions. In defining a region, there are two aspects we consider here, the invariants and the ability to distinguish a region from another region. The reason this is important here is that if a name is needed for each region, it will be part of the operation of creating the region. In addition authorization is important, but because it is important for all aspects of regions, we will address it separately in Section 2.4.

As stated earlier, one of the key aspects of a region is the set of invariants it represents or by which it is defined. In playing devil's advocate, one could argue that a simple definition with respect to invariants would be to declare that each region defines a single invariant. In conjunction with this, a key function will be operations for intersection among regions. This ignores the fact that the second central aspect of a region is a boundary, and, unless we provide some means of defining a merged boundary for the intersection, we have lost a significant aspect of the region abstraction. Hence, for the purposes of this work, we will assume that a region can have more than one invariant. We will find later that the intersection function may still be a valuable function, but it cannot replace the region. Furthermore, issues of the relationship between individual entities, the invariants of a region, and membership in the region will be discussed further in Section 2.2

The second issue we consider here is that of distinguishing regions from each other. There are many approaches taken to handling the question of distinguishing regions or groups. At one extreme the search space or region is defined simply as the set of sites that can be reached,

perhaps by some bounded multicast. This approach simplifies the issue of whether or how to declare the region within which some activity will occur, but does not allow for specifying anything beyond the multicast scope, without some additional and more complex mechanism. Another approach to defining sets of entities is to decide that all entities that share a pre-defined set of invariants are by definition in the same region. In this case, it is not difficult to specify different regions, simply by specifying the appropriate set of invariants, but if the scope of the universe is the whole Internet, then it is unlikely that one could ever have any significant level of confidence that all or most of a region's membership could be known at one time and place. Furthermore, this approach assumes a global definition space for invariants. Our position is that it is necessary to be able to identify regions distinctly. One way of achieving distinction might be to postulate that each region can be distinguished from all others by its set of invariants. This has two implications. First, there must be common agreement on invariant representation, so that two statements of invariants that are not intended to be the same as each other are distinctly represented. This implies global definition of invariants. Second, no two regions will have the same set of invariants assigned to them. This would also require global coordination. Our approach is that each region is assigned a globally unique name or identifier. There exist a number of global naming schemes[1] at present with varying degrees of scalability, user-friendliness, etc. We will choose one of these. For our purposes at present, which one does not matter, only that a region must be assigned a globally unique name.

Finally, in this section we must consider destruction of a region. The question here is whether regions are destroyed explicitly or are garbage collected automatically when they have no membership.[2] Because regions may be created as placeholders to have members included in them later, we will not support garbage collection initially, but this will need further research.

Thus to summarize, the management of a region will involve at least the following sorts of functions: **create_region**(invariants, name, …) and **destroy_region**(name, …).[3]

2.2. Region membership

The functions a region must support with respect to membership fall into two subcategories. First, there is the issue of insertion in and deletion from a region and, second, there is the issue of learning about and distinguishing among members. These two groups of functions will be discussed separately.

There are two key issues with respect to an entity joining a region, the invariants and introduction. The intention is that if the invariants were not true of the entity prior to joining a region, that they become true. The hitch here is that that may be impossible or unacceptable. If the region consists of people with green hair, for people with no hair that may be impossible and for people without died hair it may be unacceptable. So, joining may be more than a simple decision to add an entity to a region; there may be physical capability or policy decisions by or on behalf of the potential members, for example.

Because invariants are not globally defined, membership in a region cannot be based on invariants of the entities. Therefore an introduction function is needed for inserting entities into regions. In practice we may find two sorts of such introductions, those performed by a human and those performed by members of the region itself. For example, if I am defining my "home" region, I might buy a new lamp, bring it into my home, and declare to my "home" region that the lamp is now part of my region. Later, I might buy a new switch, introduce it to the lamp as the lamp's new controller, and in turn the lamp might introduce the switch into my "home" region. This is clearly making an assumption that my lamp has authority to make such an introduction, which in turn implies that it has the requisite identity and functionality. We do not expect all entities to have such a capability, but

[1] Examples are the DNS, URIs, URNs, GUIDs, and so on.

[2] We have a Master's thesis in progress on this topic, but that work is not yet complete.

[3] The precise set of functions and their arguments are part of the subject of this research. These and the other functions presented later in Section 2 are representative of what will be included in the definition and implementation.

recognize its utility. In both of these cases, an explicit introduction action will be taken, although it need not include human intervention. We recognize that key problem in scaling up of networks is to allow for removal of humans from the loop.

One can consider a similar distinction for removal or deletion from a region, but possibly with different conclusions. We assume that when an entity is assigned to a region, it inherits the invariants defining the region, and hence we could reasonably postulate that if the entity is explicitly and authoritatively assigned contradictory invariants, it is expelled from the region. It is worth noting that, in order to make this statement, we must be assuming that entities have a set of invariants, which can be modified.[4] Further, if an entity was introduced into a region by another entity already in the region, one can ask whether the introduced entity will be expelled if its introducer is removed. Here, the intuition is that that does not make sense. Consider the lamp and switch again. Just because I dispose of the lamp, does not mean that I am intending to dispose of the switch. I might choose to retain it, knowing that at a later time I will acquire another lamp, which will be controlled by the switch. Thus, we can conclude that in addition to an explicit removal function, there may be a more implicit action that may occur.

In addition to determining how an entity becomes a member of a region or is removed from a region, we must also discuss some of the other basic membership functions of a region to list or select members. Initially, we expect these to be extremely simple, and expect that the set may become richer with time. The exact behavior of these will depend on size and performance criteria as well as guarantees that may or may not be available in the implementation of the region itself. These implementation issues will be discussed further below, in addressing self-organizing adaptation. The obvious initial operations will be to list membership in the region, query about whether a particular entity is in the region, and search the region for entities that match some query. One can ask how accurate the results of these will be for a region. The accuracy may be the result of size, performance or other cost limitations, and possibly distribution of the information.

As mentioned earlier, we plan to provide several set functions across multiple regions. The one we expect to get the most use is the intersection function, although only time and experience will tell. This will provide the capability of discovering entities that fall into two or more regions simultaneously. The other obvious function to include is union. We may find others to be useful as well.

We can summarize the set of operations on membership as: **introduce_into_region** (entity_id,…), **modify_invariants**(entity_id, attribute_value-pair,…), **remove_from_region** (entity_id, …), **list**(…), **member?**(entity_id, …), **search**(query, …), **intersection**(list of region_ids, …), **union**(list of region_ids, …). All but the last two will be applied to a specific region.

2.3. Additional client functions: boundary crossings and notification

The explicit notion of the boundary of a region provides the opportunity to enable a rich functionality when activities touch or cross those boundaries. As stated earlier, a boundary is a logical concept, not bound to a particular topological or physical space. There are two aspects to the discussion about boundary management functions, the functions that take place when an activity within a region reaches a boundary and the activity models themselves.

The notion of boundary management can be made explicit by the provision of three sorts of actions: detection, modification, and notification. Detection is that task of discovering when a boundary crossing is occurring. Thus, for example, detection occurs when a packet moves from one AS [RL94] to another or from one DiffServ cloud to another [BBC98], although the region abstraction allows for an infinite variety of

[4] This may appear to be a contradictory statement, since invariants should be immutable. In fact, what we are saying here is that an invariant may be applicable to an entity only for limited time periods. An example of this might be the lamp, which I bought originally, but then gave to my friend. The invariant statement about ownership may change with time.

boundary definitions. When this boundary detection occurs, in some cases, state will be changed. This may occur as the change of a field in a packet, a charge being incurred by counting the packet or its size, or, again, one of a large variety of other changes. Finally, as the detection occurs the appropriate response may be some explicit notification. Part of the issue with respect to the notification is the selection of the recipient of that notification, determined at least in part by the nature of the activity in progress.

We find a large number of differing activity models, each suggesting a set of places notification might be sent, based on the locus of control of the activity. At one extreme there is the packet that is moving through the net. Another related, but much richer model is that agents (see for example the work of Minar, Kumar, and Maes [MKM99] or Tripathi and Karnik [TK98, TK99]) are moving through the net. The packet moving through the net can be considered a small amount of state being transported. In contrast the agent can be considered to be much more state in conjunction with one or a set of processes on the move. A third model is that of remote invocation, in which an activity is occurring remotely, but final control and state are being maintained at some static point in the net. Yet another model is one of continuations. This can be viewed as a middle ground between a remote procedure call in which responses return to the invoker or static point and the agent model in which a very rich entity is moving through the net. The continuation model can be considered more of a lightweight thread moving through the net.

Now consider the locus of control or at least authority and hence possible destinations for notification. In every case, one candidate recipient is the transit point, because it is there that a decision may need to be made about whether to support transit or not. Additionally, in the case of the packet, control or authority may reside with either the sender or receiver. In the case of the agent, the recipient of the notification is likely to be the agent itself. For remote invocation, the invoker is a likely recipient, and in the continuation situation, either the thread itself or only the transit point. Other paradigms may also present themselves, and the region must be prepared for any reasonable destination of the notification. In other words, our challenge is to develop a single abstraction that can cleanly and efficiently support all of these models.

The functions described in this section will not be reflected as specific operations on regions in the same way as in the previous sections. Execution of functions or movement of agents to or from locations will be specific to particular entities within the region not the region itself. The notification may be something that is triggered in the region itself or the boundary element.

2.4. Security

As stated at the beginning of Section 2.1, we have sidestepped the issue of security in the previous subsections above. This is a topic that absolutely must be part of an effort such as this. There are two key aspects to security with respect to regions: the integrity of a region itself, and the privacy or other security it provides to others. For regions to be useful, it must be possible to trust their membership to be accurate and authorized. For example, if company X wants to create a region containing all of its network end-points and only its network end-points in order to build a private overlay network, it needs to be able to trust that the integrity of that region will not be compromised. Such compromises may cause problems ranging from exposure of private information to denial-of-service attacks. The second problem is providing trustworthy reflections of boundaries. By this we mean that if an agent or packet is moving through the network, it is important to be able to know in a trustworthy way whether or not a region boundary trigger is or will be occurring. The first is an issue for members of the region and the second for the clients or users of the region.

2.5. Comment

It is important to remember that the intention of this work is that regions will be capable of supporting a wide variety of functions and objectives. In this section we have considered mechanism, but have intentionally stayed away from explicit purpose, in order to enable use for a variety of purposes.

There is lurking in here a deep and challenging problem. Many of the activities we are considering are not activities of the region itself.

They are activities that involve individual members of a region or set of regions. Thus, for example, moving packets or mobile code among elements of a region or across region boundaries or executing transactions within a region or across region boundaries are not activities of the regions themselves, but activities of the elements of the regions. At the same time, we are postulating that the region entities involved will take action under certain conditions, and, in fact, will be monitoring the situation in order to recognize those conditions. We can postulate that it is necessary to cleanly separate the region abstraction from the abstractions of the elements, yet provide the intertwining of these activities.

3. Adaptive Regions and self-optimization

As mentioned in the introduction, regions will have the capability of re-organizing themselves in order to improve their behavior. Again, this is something that will be triggered, although in this case the trigger will be based on a set of criteria that may involve, size, patterns of usage, demands for performance, other costs, and how all of these can and should be traded against each other. This occurs "below the abstraction", at the implementation level. It is the mechanism that allows a single abstraction to meet a wide range of performance and scalability requirements.

It is important to note here that re-organization may have an impact on the degree of accuracy that can be achieved by a function. It will be necessary that the definition of region functions include the ability to factor in degrees of accuracy.

Depending on circumstances, a region entity may be implemented by methods ranging from a simple centralized server to a globally distributed computation over widely dispersed, possibly replicated information. It is important to discuss how and when a region entity might transparently improve its performance when circumstances change. There are at least three aspects that may lead to the decision to re-organize or optimize a region: size, usage patterns, and distribution of the members and clients. Improving the situation with respect to one or another of these aspects must lead to careful consideration of whether it will improve or worsen the situation with respect to another aspect. In each case, as an evaluation takes place, there must also be a consideration of the cost of a transformation. This process of evaluation will be complex, particularly because the overhead must also be kept as low as possible.

Finally, the overhead on the clients must be kept to a minimum. Thus, for example, if a re-organization were to require that any requests in progress be re-submitted, this might be a problem for clients. More importantly, it is possible that distribution or redistribution of a region's representation could cause the degree of reliability of some operations to change. Thus, the whole process of evaluating the current representation and behavior of a region and whether or not it should be transformed in some way or another must be carefully designed. A framework for choices about optimization of regions is necessary to enable adaptation of a region to evolving conditions.

3.1. Size

When a region is small, the size of the representation of its membership probably does not matter. As a region grows, its representation may become increasingly cumbersome. If size itself is an issue in supporting a region, then more efficient representations may become valuable. Converting the structure of a region from one representation to another will incur some cost, simply in performing the transformation, so the choice to make such a conversion must include evaluation of that additional cost.

The issue of when to re-organize is amenable to both a simple approach as a starting point, with, later extension to more sophisticated approaches. The simple approach is to provide fixed values for hysteresis; this would mean that the critical point for re-organizing during growth would be larger than the critical point for re-organizing during shrinkage. The first step in making this more sophisticated might be to vary the difference between these critical points depending on a history of transitions across those points. Additional sophistication may be achieved by considering other costs of re-organization, as well as other broader effects. In another sort of approach, either the cost of re-organizing might be

spread out continuously, or cost might the farther one gets from the boundary conditions.

3.2. Usage patterns

There will be a number of different ways a region may be used, reflected in the list of functions described in Section 2. Depending on the frequency of each of these sorts of operations, different organizations of the region may improve or worsen the situation. If the membership in a region is extremely dynamic, then insertion and deletion should be efficient. If modifications to the information about elements occur frequently, then that should be optimized. With respect to use of the elements of the region, if listing membership dominates over selecting individual elements, that should be made efficient, and so on.

Both absolute numbers of the different kinds of actions and the relative balance among the kinds of actions may be important. If all usage of a region is quite low, then it probably is not worth re-organizing at all. As with size, hysteresis may play an important role here. Once a decision has been made to re-organize in order to improve performance, it should take into account the balance of usage, not just that a particular sort of usage needs improvement.

3.3. Distribution

A third criterion we call *distribution*. The question that must be asked here is the extent to which either partitioning[5] or replication of the representation of a region will improve apparent performance. If the clients of a region are widely distributed topologically, there may be several reasons to distribute the infrastructure representing the region. For example, if network access is either low-bandwidth or quite variable, then placing some of the infrastructure closer to the clients may improve apparent performance. If the patterns of usage can be partitioned based at least to some degree on this topological distribution of clients, the infrastructure might be partitioned. In contrast, if the usage cannot be separated well by topology, or if usage causes a great deal of secondary traffic among the other parts of the region, replication may be a more desirable model for organization. As mentioned earlier, an extreme example of this is one in which operations become increasingly unreliable with replication or partitioning. Thus, in considering distribution and partitioning not only the clients' distribution but also the isolation or integration of a region may have an impact on whether and how a region's infrastructure is distributed.

[5] We use the word "partitioning" in the sense of a database, not in the sense of a network.

3.4. Making decisions

The decisions about adaptation must be made not only to accommodate all these issues, but also in such a way that the clients' perceptions of a region is that it is not too frequently in flux or too difficult to access. In addition, the cost of adaptation must always be considered, in addition to the cost of managing the infrastructure both before and after re-organization. One result of this component of the region infrastructure must be a framework for decision-making about adaptation

4. Examples of uses

With the above discussion in mind we can explore several scenarios in which regions provide capabilities not otherwise available. In the first, we explore an improvement in network overlays, using regions as the basis. The second example demonstrates the use of notification of a boundary crossing in order to support billing at multiple layers. The third section addresses the use of regions for building and managing network based applications.

4.1. Application layer overlays

In this section, we explore the use of regions defined at the network layer as a vehicle for optimizing and improving application layer services. In this example, the lower level is unaffected by the higher level, but its information enables more informed decisions and operation at the higher level.

A topic of great interest today is building overlay networks. These are typically sets of hosts or end-nodes from the perspective of the packet-level Internet that provide a network of infrastructural components for some application. For example, web cache servers provide information sources for Web users. In many cases the cache servers are hosts from the perspective of the Internet, but

from the perspective of the browser they are an invisibly embedded in the net.

Now consider the problem of some application level request. There may be a number of possible routes that such a request could take through the overlay network. Some choice is made at the application level about that path, but because the application level has no information about lower level routing, that decision cannot be made based on actual paths followed. The traffic may traverse the same links many times before actually achieving the desired goal. Choices at the application level imply that more than one option was possible and perhaps having lower level routing information could change or at least better inform such a decision. For example, both Tapestry [ZKA01] and Pastry [RD01] depend on routing tables, in which there is choice about the entries. In their basic schemes at best these schemes may use some roundtrip time (based on ping or similar roundtrip measures) to make choices about proximity for entries into the routing tables. But these do not avoid retracing steps. Nor do they support any ability to make policy choices about points in the routes selected.

At the routing level, we already have some very useful regions defined, known as routing domains or autonomous systems (AS's). These regions define their invariants as those destinations recognized by the routing algorithms as having the same gateways into the region, generally based on subnet masks of the IP address. Their entry and exit points are well defined, again the gateways. If we add to the region information something about the application level services provided by elements in those regions, it is possible that the application, deciding on a route through an overlay network can now make more intelligent decisions about the route to be used.

One option this approach allows is keeping traffic in the overlay network within an AS. There may be a variety of reasons for that decision such as cost, performance, or privacy. Since an AS is often the definition of a corporate boundary, a corporation may prefer the idea that generally the traffic of its applications remain within its AS. The enhanced region reflects not only the IP level boundaries of the corporation, but also information about how the application overlay can stay within those boundaries. The key point is that this is just one example. The power of the region abstraction is that alternative regions can be defined by the use of invariants other than those arising from AS's, in order to define new routing regimes.

One can imagine a more complex, but related example in which again each AS reflects a corporate boundary, but now rather than requiring that traffic stay within one such boundary, a corporation may have a set of priorities. The preferred option is to stay within itself. But if that is not possible, there may be an ordered preferential list of alternatives. Such a set of regions may include the elements of a variety of application overlay network elements, for different applications or application suites. A single set of region definitions may serve many applications, helping to make routing choices for each using the same set of corporate criteria such as cost, efficiency or privacy policies. Within each such AS there may be an application network overlay router.

4.2. Crossing region boundaries

By viewing a region as a boundary with controlled crossing points, we can place functionality that is necessary and possibly shared at those boundary crossings. The sets of entry points and exit points will be subsets of the total membership of the region. It is likely that the smaller those sets are the more likely they will be amenable to centralized or at least coordinated control. It need not be the case that an entry point is also an exit point or vice versa. They can be co-located but need not be.

Consider an agent that is gathering information on behalf of its owner. As it moves through the net it accretes information. There are several examples of charges for which it may be responsible. Let us postulate that each potential source of information sits within a region. Each source of information or group of sources within a region has some billing policy. In some cases, each information source charges for each piece of information it provides to the agent. In other cases, there may be a flat fee for as much information as the agent wants. In yet others, there may be some group charging, so when an agent arrives at one information source it receives an admission ticket

for all information sources within the region for a fixed fee. Other billing models may also exist. In addition, regions themselves may have transport billing. Some charge per bit for traveling along their links, while others may charge a flat fee for travel within the region. Some have charges for bits transiting their boundaries in or out. Again, a variety of policies are possible. The key that is important here is that there may be charging for at least two sorts of service, moving bits around and provision of information.

In this example, it is important not only for the transit points and information servers to know when an agent is arriving or departing, possibly whether it has been there before and so on, but the agent itself may care, if it is trying to minimize cost The agent is moving around collecting information. It must consider fees at one or a set of information sources, fees at region boundaries, and fees based on its size. Hence it also wants to be notified of potential transitions. It is likely to know, without notification when it is moving from one information source to another, but potential border transition may be something to which it will need to be alerted. In this case the region boundary may cause the agent to change its plans. Hence this example allows us to explore the relationship between activities occurring in conjunction with specific members of a region and notification as provided by the region itself.

We recognize that there are possibly fatal problems with deploying agents, especially having to do with security. There has been significant work on this including [MKM99, TK98, TK99]. To address this, one can provide similar functionality using one or another form of portable code, with its own set of security problems. One issue is the extent to which they are self-directed and collecting state they carry with them.

4.3. Creating mobile applications in a pervasive environment

A third example allows us to explore an even richer set of boundary crossing issues, interactions among multiple regions, and questions about expanding activities to more regions as needed.

Visions of the future of our computing environment suggest a broad base of fixed computing devices and services, the pervasive computing environment, through which will be moving humans and other entities, each of which may be served by a suite of small mobile devices. This is one form of the vision of Weiser [We93] for the ubiquitous computing environment. We can assume that only those devices that have some connectivity to others (networking capabilities of some sort) are of interest here. Furthermore, it is increasingly likely that at least some of those devices will be capable of using more than one network technology.

Now, consider the problem of creating applications in this environment. The application will no longer run on a single monolithic workstation. Instead, its user interface devices may at different times include watch displays, wall displays, printers, speakers, headphones, haptic devices of various kinds, pointing devices, keyboards, microphones, and so on. In addition, as suggested, some of these devices may be capable of using several different network technologies. In this situation, one must consider not only differing network technologies, but also changes in the environment, changes in the management policy of elements, in addition to the obvious set of problems arising from mobility.

Consider the following simple scenario. We postulate a new kind of activity we call *catalysis* that will cause an instance of an application to come into existence. How does this happen? A *catalyst* will contain a set of objectives or functions that the application must provide, as well as a set of requirement specifications for components needed to realize the application. These may include devices of certain types, but they may also include network resources such as transmission capabilities, caching, or whatever, in addition, to more ephemeral elements (objects) that provide certain functions. Examples of these might be implementations of specific encryption algorithms, particular sorting algorithms, a transaction manager, and so on.

In addition to the requirements of the catalyst itself for forming the application, there are two other sets of requirements and constraints to be considered, those of the user and those of the potential elements of the application. The user may have both functional and policy

requirements, such as which algorithms are acceptable, configuration of devices, acceptable vendors of service, or prices ranges. The elements may have security or loading constraints, pricing and so on. It is important to recognize all three aspects of catalysis, going beyond the more common dynamic configuration that reflects only acceptable functional composition.

In order to limit the search and discovery of acceptable resources and components of the application, it will be invaluable to be able to identify the set of regions (one or more) to be used. Thus, for example, if the catalyst is building a home alarm application, it will be valuable to limit the catalysis to the homeowner's home, the neighborhood (for notifying neighbors perhaps) and the appropriate municipality (for notifying the policy or fire department). By considering an example in which multiple regions are central, we are able to explore the relationships among regions, as well as questions about implicit vs. explicit nesting. For example, one can ask whether it is valuable to consider everything in my home region to be part of my municipality region or whether keeping these two concepts and hence regions distinct is more effective. This particular example was also chosen to allow for the fact that a neighborhood may span more than one municipality, hence allowing us to explore overlapping but not nested relationships among regions.

For making decisions for catalysis it may be valuable to include elements from various different layers of abstraction in a single region, in order to make the most effective choices. For example, various different devices (siren, telephone, etc.) may be connected into the home network using base stations supporting different technologies. So, a device that was being considered as part of a fire detection system would be more useful on a wired or radio based network than an infrared connection. In a more sophisticated decision process, a route to a device that included an IR link would be less desirable. So, if the region could actually capture information about the elements of the network and connectivity, in addition to the obvious first level of resources the application may need, the region will be a more useful utility for catalysis.

In addition, it may important for traffic that leaves the home region be authenticatable, unforgeable, and private. Privacy is particularly challenging because it may be important not only that individual messages not be readable, but also that the fact that the police are being notified of a burglary is masqueraded.

This example allows us to explore a number of challenging aspects of regions, including the relationship among overlapping or nested different regions such as union and intersection, naming regions, in addition to a key set of questions about the multi-layered role that regions may play. An example such as this also demonstrates the need for regions as mandatory bounds on some activities, as locale of notification in other cases, and as point of transition in yet others.

5. Related work

The related work falls into several major categories, partitioning of namespaces in order to handle scaling of name assignments and resolution including discovery for peer to peer applications, boundaries defined in order to reflect changes in some activity, cross protocol layer interaction, agent technologies and their security problems, middleware infrastructure to support creation and execution of network based applications, and infrastructure adaptation. We will only be able to provide a sampling here.

One set of problems is grouping of objects in order to address scaling problems. In each of these examples, the problem was to reduce the space to be searched in order to find something. The sole function of the Domain Name System [Mo87a, Mo87b] is to provide a single global hierarchy in which both name assignment and name resolution occur in order to find hosts. CORBA [Ob00, Ob01] provides a much richer set of middleware activities, but in conjunction with this provides a two level hierarchy for naming, by uniquely naming each ORB, and delegating unique assignment with the ORB to the ORB itself. Here the objective was to find a specific CORBA object. The Intentional Naming System [ASBL99] was designed to route traffic to the named entity. It is an example of a different approach, in which names are composed of

attribute value pairs, but these are organized hierarchical. An entity announces itself to any resolver, which in turns broadcasts the identity of the entity using a spanning tree to the universe of resolvers. A request to the entity is resolved and forwarded at each resolver between the requester and the entity itself. Although this work as it stands does not scale, it provides an interesting point within the space of naming alternatives, because it attempted to provide multi-layer functionality of both naming and routing.

A new and evolving topic is how to find elements in support of peer-to-peer communication. Ingrid [FKSS95] was an early attempt to address the problem. In Ingrid each entity is identified only by an unordered set of attribute value pairs. Grouping is intended to be global, based on matching sets of attribute value pairs. From one perspective, a fatal problem with this approach is that one can never know whether all the elements of a group, those sharing attribute value pairs have discovered each other, because Ingrid is completely decentralized. Gnutella [Cl01] takes a very different approach. It is also intentionally completely decentralized, but the approach to scaling is to limit each search the peer group within a searcher finds himself or herself in at any given time. This is the bounded multicast approach, with little control over the starting point of the search. In addition to Ingrid and Gnutella there is an increasingly large number of approaches to peer-to-peer communication including BEA WebLogic [BE99, BE01], IBM's WebSphere [BM00], Sun's JXTA [Go01], and the Universal Description, Discovery and Integration Team [Un00], as well as structured peer-to-peer systems such as Tapestry [ZKA01] and Pastry [RD01]. Naming and discovery are important because we will need to be able to discover regions and have postulated that it will be based in part on naming.

Almost any horizontal slice through the current Internet structure reveals a loosely couple federation of separately defined, operated and managed entities, interconnected to varying degrees, and often differing drastically in internal requirements and implementation. We find two specific examples in routing and QoS provision. BGP [RL94] boundary transitions reflect routing protocol changes between BGP used between AS's and one of a variety of internal routing protocols. DiffServ clouds [BBC98] reflect boundary points at which per domain behavior may change. In each case the choices of what happens internally to a region or scoping entity are made independently of what is happening outside. Wroclawski [Wro98] proposed that a key architectural component of the *Metanet*, a new, multidimensional network or networks, is a *region*, quite similar to ours.

In terms of middleware support for the creation and support of distributed applications there is an enormous collection of work, including CORBA [Ob00, Ob01], Microsoft's Universal Plug and Play [Mi00], Sun's combination of Jini [Su00, Ed01] and Rio [Su01], and the W3C's combination of XML [BPSM00, BHL99], RDF [BG00, LA99], and SOAP [BE00]. This is an area where a great deal of work is occurring, so this is just a sampling of the activities. In the work on support for creation of applications, we intend to build on existing work. We will evaluate the various options, but the tools available from Sun appear to be a good starting place.

Although we explore the potential relationship between agents and regions (see the work of Minar, Kramer and Maes [MKM99], as just one example of a great many on agents), we recognize that there is an ongoing problem with security with respect to agents. Tripathi has explored this and is demonstrating his approach in the NSF funded Ajanta project [TK98, TK99]. As much as possible we intend to build on top of existing work, so we will use something like Ajanta, which is implemented in Java.

Finally, there is related work in the area of adaptation. Much early work came from the algorithms community and was collected in such textbooks as the work by Corman, Leiserson and Rivest [CLR90], which provides, as an example, interesting algorithms for managing B-trees under a variety of constraints. In some cases, the algorithms reflect one-time significant costs, and in others repeated smaller costs. Such tradeoffs must be part of our consideration. There has also been a significant amount of work from the database community. There was a flurry of work about 10 years ago, and a reviving of interest in

the last few years, including a review of mechanisms within IBM [SI96] and as an example the more recent paper by Zou and Salzberg [ZS98]. We expect the database community to be an extremely useful source of adaptation algorithms and cost analysis, since they are generally constrained by real implementations and real customers.

6. Conclusions

Consider each type of invariant as one dimension, then a region is a bound part of a multidimensional plane defined by the types and values of the invariants for that region. Then want to understand the different planes that intersect at each member, in order to learn about different sorts of regions to which each member belongs. Transitions from one region to another are another way of describing the "waypoints" defined by Wroclawski [Wro98] in his Metanet model.

In this paper we have proposed the concept of the **region** be a key design element in an architecture for extremely large scale, widely distributed, and heterogeneous networks, as a mechanism for grouping, partitioning, and formalizing boundaries around those groups and partitions. We propose that common use of regions will enable increased functionality and management. Furthermore, we propose that by providing a general mechanism, increased capability is made available in each use, in part due to our ability to provide a generic framework for adaptability that is generally not provided in any individual instantiation of a similar mechanism.

References

[ASBL99] Adjie-Winoto, W., Schwartz, E., Balakrishnan, H. and Lilley, J., *The design and implementation of an intentional naming system*, **17th ACM Symposium on Operating Systems Principles (SOSP '99)**, *Operating Systems Review*, **34**(5), December, 1999, pp. 186-201.

[BBC98] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W., **An Architecture for Differentiated Service**, RFC 2475, Internet Engineering Task Force, December, 1998,

[BE99] BEA Systems, **BEA WebLogic Enterprise Introduction**, Document edition 4.1, BEA Systems, May, 1999. Available aa hppt://www.bea.com/products/weblogic/enterprise/enterprise_intro.pdf.

[BE00] Box, D., Ehnebuske, D., Kakivaya, G, Layman, A., Mendelsohn, N., Nielson, H. F., Thatte, S., Winer, D., **Simple Object Access Protocol (SOAP) 1.1**, W3C Note, May, 2000. Available as http://www.w3.org/TR/SOAP/.

[BE01] BEA Systems, **Making Component-based Systems Scale with BEA WebLogic Enterprise**, BEA Systems, undated/ Available as hppt://www.bea.com/products/weblogic/enterprise/papers.html.

[BG00] Brickley, D., Guha, R. V., **Resource Description Framework (RDF) Schema Specification 1.0**, World Wide Web Consortium, March, 2000. Available as http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[BHL99] Bray, T., Hollander, D., Layman, A., **Namespaces in XML**, World Wide Web Consortium, January, 1999. Available as http://www.w3.org/RF/REC-xml-names/.

[BM00] Beck, B., McGinnis, M., **IBM WebSphere Everplace Suite v.1.1 White Paper**, International Business Machines, October, 2000. Available as http://www-3.ibm.com/pvc/products/pdf/wes.pdf.

[BPSM00] Bray, T., Paoli, J., Sperberg-McQueen, E. M., **Extensible Markup Language (SML) 1.0 (Second Edition)**, World Wide Web Consortium, October, 2000. Available as http://www,w3,org/TR/REC-xml/.

[CLR90] Corman, T. H., Leiserson, C. E., Rivest, R. L., **Introduction to Algorithms**, MIT Press/McGraw-Hill, 1990.

[Cl01] Clip2, **The Gnutella Protocol Specification v0.4, Document Revision 1.2,** undated. Available as http://www.clip2.com/GnutellaProtocol04.pdf.

[Ed01] Edwards, W. K., **Core Jini**, **Second Edition**, Sun Microsystems Press, Prentice-Hall PTR, 2001, ISBN 0-13-089408-7.

[FKSS95] Francis, P., Kambayashi, T., Sato, S., Shimizu, S., *Ingrid: A Self-Configuring Information Navigation Infrastructure*, **4th International World Wide Web Conference**, December 11-14, 1995, Boston, MA, USA, pp. 519-537. (Also available as http://www.ingrid.org/francis/www4/Overview.html.)

[Go01] Gong, L., *JXTA: A Network Programming Environment*, **IEEE Internet Computing Online**, June 27, 2001.

[LA99] Lassila, O., Swick, R., **Resource Description Framework (RDF) Model and Syntax Specification**, World Wide Web Consortium, February, 1999. Available as http://www.w3.org/TF/REC-rdf-syntax/.

[Mi00] Microsoft Corp., **Universal Plug and Play Device Architecture, Version 1.0**, June, 2000. Available as http://www.upnp/org/download/UPnPDA10_20000613.htm.

[MKM99] Minar, N., Kramer, K., Maes, P., **Cooperating Mobile Agents for Dynamic Network Routing, Software Agents for Future Communication Systems**, Springer-Verlag, 1999, ISBN 3-540-65578-6

[Mo87a] Mockapetris, P., V., **Domain Names – concepts and facilities**, RFC 1034, Internet Engineering Task Force, November, 1987.

[Mo87b] Mockapetris, P. V., **Domain Names – implementation and specification**, RFC 1035, Internet Engineering Task Force, November, 1987.

[Ob00] Object Management Group, **Discussion of the Object Management Architecture (OMA) Guide,** Object Management Group, Doc. Number 00-06-41, 2000. Available as http://www.omg.org/cgi-bin/doc?formal/00-06-41.pdf.

[Ob01] Object Management Group, **The Common Object Request Broker: Architecture and Specification**, Rev. 2.4.2, Doc. Num. 01-02-33, February, 2001.

[RD01] Rowstron, A., Druschel, P., *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*, **Proc. IFIP/ACM Middleware 2001**, March 2001.

[RL94] Rekhter, Y., Li, T., **A Border Gateway Protocol 4 (BGP-4)**, RFC 1654, Internet Engineering Task Force, July, 1994.

[SI96] Sockut, G. H., Iyer: B. R., *A Survey on Online Reorganization in IBM Products and Research*. **Data Engineering Bulletin 19**(2), 1996, pp. 4-11.

[Su00] Sun Microsystems, **Jini ™ Technology Core Platform Specification, v. 1.1**, Sun Microsystems, October, 2000. Available through http://www.sun.com/jini/specs/.

[Su01] Sun Microsystems, **Rio Architecture Overview**, White paper from Sun Microsystems, March, 2001. Available as http://www.sun.com/jini/whitepapers/rio_architecture_overview.pdf.

[TK98] Tripathi, A., Karnik, N**., Resource Protection in a Mobile Agent System**, Technical Report 98-011, Dept. of Computer Science, University of Minnesota, Twin Cities, 1998.

[TK99] Tripathi, A., Karnik, N., Vora, M., Ahmed, T., Singh, R., *Mobile Agent Programming in Ajanta*, **Proceedings of the 19th International Confernce on Distributed Computing Systems (ICDCS '99).**

[Un00] Universal Description, Discovery and Integration Team, **UDDI Technical White Paper**, International Business Machines Corporation and Microsoft Corporation, Sept., 2000. Available at http://www.uddi.org/whitepapers.html.

[We93] Weiser, M. *Some Computer Science Issues in Ubiquitous Computing*, **Communications of the ACM**, **36** (7), July, 1993, pp. 75-84.

[Wro97] Wroclawski, J., *The Metanet*, **Research Challenges for the Next Generation Internet**, ed. Computing Research Association, May 14-17, 1997.

[ZKJ01] Zhao, B., Kubiatowicz, J, Joseph, A. *Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing*, UCB Tech. Report UCB/CSD-01-1141, UC Berkeley, April 2001.

[ZS98] Zou, C., Salzberg, B., *Safely and Efficiently Update References During On-line Reorganization*, **VLDB'98, Proceedings of 24th International Conference on Very Large Data Bases**, August 24-27, 1998, New York City, New York, USA, Morgan Kaufman, pp. 512-522.