RZ 3252 (# 932987) 07/03/2000
Computer Science/Mathematics 9 pages

Research Report

The IcorpMaker: A Dynamic Framework for Application-Service Providers

Sean Rooney

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

The IcorpMaker: A Dynamic Framework for Application-Service Providers

Sean Rooney
IBM Research, Zurich Research, 8803 Rüschlikon, Switzerland
email: sro@zurich.ibm.com

Abstract

Application Service Providers (ASP) simultaneously host the commercial activities of many clients. The expectation is that the number of clients that a large ASP will support will increase by several orders of magnitude as the market for service hosting expands. Being able to automate the process of adding new clients and dynamically modifying the resource allocation of existing ones has obvious advantages. However, before this can be achieved a set of significant technical problems needs to be addressed in relation to the dynamic resource allocation across the ASP's infrastructure. In such a demanding environment ASPs cannot know the nature and constraints of all their clients' commercial activities. On the other-hand, clients should not be expected to know much about the infrastructure upon which their activity is being hosted.

A framework—the *ICorpMaker¾* is described which allows much of the intelligence for resource allocation within an ASP's infrastructure to be delegated to the clients themselves, without requiring clients to know precise details about that infrastructure. This allows client control software to increase or decrease a client's current resource allocation in response to client specific information.

1 INTRODUCTION

In an environment in which several different entities are competing for resources, a given entity can be sure of getting enough resources at a specific time if either resources are so plentiful as to exceed maximum possible demand or that entity can reserve resources usage in advance. The former is simpler, but more wasteful, where waste is defined as the amount of unused resources over a given period.

A company leasing infrastructure to third parties in order to support some commercial activity is often termed an Application Service Provider (ASP). ASPs must decide if it is sufficient to simply make resources plentiful in order to address the contention problem or whether some reservation mechanism is required. There is no general reply to this question as it depends both on how much resources cost and how much clients are willing to pay; however, clearly an ASP that can offer the same service to a client at a lower cost (due to the fact that the ASP suffers less waste and therefore requires less infrastructure) has a competitive advantage.

Our assumption is that ASPs will choose to partition the infrastructure between clients in order to give them resource guarantees. We further assume that the number of clients a large ASP will be required to support is such that the process of introducing new clients must be almost entirely automated. This leads us to conclude that, in the long run, ASP's infrastructure must allow clients to specify what resources they want and allow them to change their allocation over time dynamically.

One approach would be to require clients to specify exactly how much low-level resource they require, e.g. bandwidth, router buffer, server CPU cycles, etc. In order to determine the correct allocation for a desired system behavior, clients would need to have an extremely detailed knowledge of both the nature of the ASP equipment and the topology of their network; in general such information is not available to clients and making it so would render the infrastructure difficult to evolve. Moreover, even if one knows the precise capacities of all the equipment and the different technologies used to support QoS guarantees, it is still not easy to predict the end-to-end behavior of an application knowing only the low-level resources allocated to it on that equipment. Another approach is to label some predefined resource allocations, e.g., *gold*, *silver*, *bronze*, and allow clients to choose the required level of service. However, neither the client nor the ASP has any way of knowing whether the chosen level of service is adequate for the client's needs or whether it will remain so as those needs vary over time. As the number of clients that an ASP supports grows the likelihood that all or most of them will be satisfied with a small fixed set of preprepared allocations diminishes.

An allocation does not have to be initially correct if it can be adjusted in a controlled way such that it enables client applications to behave as required. This allows an adequate resource allocation at a satisfactory cost to be found at the start of the applications execution and for this resource allocation to be modified over time as a function of changing circumstances. However, only clients know what words such as 'required', 'adequate', 'appropriate' and 'changing circumstances' mean when applied to their applications. An analogy can be drawn with a thermostat controlling the temperature of a house, the client does not have to know in advance what temperature they want the house to be, only that when it is too cold it can be made warmer and vice versa.

This paper proposes a new approach for allocating resources within an ASP's infrastructure based on delegating much of the intelligence for resource allocation from the ASP to the clients. Clients can then take into account their client-specific knowledge in order to obtain the required level of service without requiring them to have a detailed knowledge of the nature of the physical infrastructure. Achieving this type of delegation requires a means of dynamically and safely creating and modifying resource allocations on both networks and servers; the enabling `ICorpMaker` framework therefore encompasses them both.

First the issues involved in resource allocation within an ASP's infrastructure are considered. Then the `ICorpMaker` is explained through a description of a working implementation.

2 RESOURCE ALLOCATION

In this paper, content owners purchasing infrastructure from an ASP are termed *clients*, while the population that then avails itself of that hosted content are called *users*. An ASP hosting services on behalf of clients typically owns a set of servers and network. The network (or networks) interconnects the servers both with each other and with the entry points through which customers access those servers. Server and network resource allocation must be coordinated in order to give clients meaningful guarantees, for example, increasing the bandwidth available to users connected to a server may not improve the efficiency with which they obtain information from that server, if it were already overloaded.

This section describes the nature of resource allocation within an ASP's network and servers. It outlines the mechanism that allows this allocation to be coordinated and then describes how the control of this mechanism can be delegated to clients such that

they make use of client-specific knowledge in order to optimize their resource usage.

2.1 Network Resource Allocation

Although at any given time the network may be under used, the times when diverse clients need resources are likely to be tightly correlated. For example, different clients offering an internet-based stock trading service are all likely to experience increased activity at about the same time. The traffic carried over the ASPs network is therefore likely to be very bursty and some resource allocation/charging mechanism must be used to determine which traffic is carried and which dropped when the required capacity exceeds the actual one.

Network resource allocation can be achieved at different levels of granularity, for example an ATM virtual connection permits a single flow to be allocated a very precise resource guarantee, while within the IP Differentiated Service architecture [1] a coarser guarantee is given for an aggregated set of users.

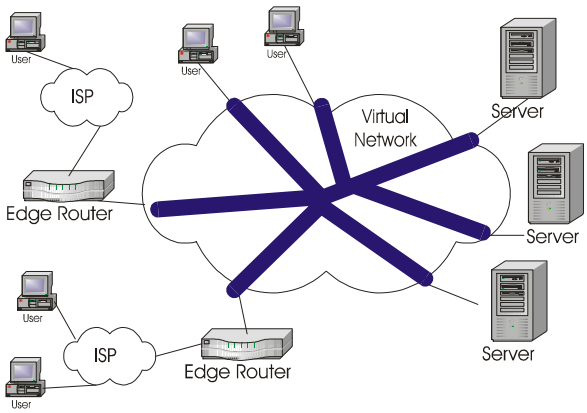

Figure 1 Virtual network.

Clients purchase network resources from ASPs in order to obtain guarantees for the traffic carried to and from multiple access points in the ASP network to multiple servers. The network resource allocation can be thought of as a virtual network. Figure 1 shows a schema of a virtual network, supported over an ASP infrastructure, which interconnects three physical servers with two edge routers and some users directly connected to the ASP's network. The virtual network appears to the client as a dedicated physical network. Many frameworks have been proposed for supporting virtual networks [2–4].

The virtual network abstraction is independent of the technology used to actually support it, e.g., a guaranteed connection rate of 10 Megabytes between A, B and C is independent of whether, this is achieved using an ATM virtual circuit, IEEE 802.1q Ethernet VLAN [5] or an MPLS tunnel [6]. Just as an IP network is supported over multiple different layer-2 networks, so should a virtual network be supportable over multiple different resource allocation mechanisms. This fact is obscured in the literature because the goal—the privileging of certain user traffic by segregating it from other traffic—is tightly bound to the means of achieving it, e.g., the differentiation of traffic at routers based on the bits in the type of service field of the IP packet.

A virtual network (`VN`) can be created with minimum assumptions about the underlying technology through a distinct control plane, in which control messages are carried between network elements and in which the mapping between the abstract notion of the `VN` and the actual underlying technology is carried out. This single abstraction allows the very different semantics of QoS present in, for example, ATM, Ethernet [5] and differentiated service [1] to be handled within a single coherent framework. Section 3 explains how the `VN` abstraction is supported within the `ICorpMaker`.

2.2 Server Resource Allocation

ASPs run applications on behalf of third parties. These can either be standard applications, e.g., an *http* daemon distributing client supplied content, or a client supplied application, for example, a new distributed game server. A server's resources are partitioned so that malicious clients may not deny service to other clients of the same server by consuming a surfeit of those resources.

Common operating systems such as Window's NT and the various flavors of Unix are designed with the principle that tasks can only be started by trusted users and that they are in general well behaved. Allowing foreign users to start arbitrary tasks on the server means that this assumption no longer holds and additional mechanisms must be found. Three approaches can be distinguished:

- extending existing operating systems [7];
- building new operating systems appropriate for running foreign code [8];
- running multiple instances of operating systems on the same physical server [9].

Our assumption is that a sufficiently large ASP will have a mix of different types of servers with potentially different means of ensuring the secure resource-constrained execution of third-party code; either a mix of different types of servers will be chosen for use in different circumstances, or an initially uniform infrastructure will become heterogeneous as parts of the system are enhanced while other legacy parts are retained.

The situation is then analogous with that of the network described in Section 2.1: the different means of achieving server partitioning also needs to be en-

capsulated in a unifying abstraction independent of the precise mechanisms being used. The obvious term for such an abstraction is a virtual server, however this term is already widely used in the literature with a variety of different meanings. As it is our intention to make use of the existing solutions of server partitioning within the framework, to avoid confusion a new term is introduced: an Instant Server (IS). An IS is defined as a set of Instant Server Chunks (ISC), where each ISC is some partition of a physical server. If a server possesses no partitioning mechanism then the entire server is one single invariant ISC.

Within the framework client's applications are run on one or more ISC. The client's ISCs and the access points of the ASP network are interconnected by a client-owned VN. IS Proxies are run at the access points in order to determine which ISC should support a given end-user request. Figure 2 shows a schema of an instant server supported by an ASP's infrastructure, in which three ISCs are interconnected with each other and with two IS Proxies over a VN.

It is convenient to couple the notion of a client's server partitions and the network partition that interconnects them into a single concept: the Instant Corporation or ICorp for short. The framework—called the *ICorpMaker*¾ supports the dynamic creation of ICorps on behalf of clients across an ASP's physical infrastructure.

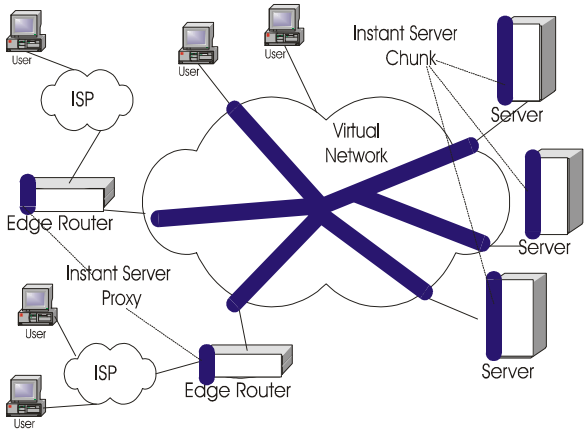

Figure 2 Instant server.

At the start of operation, clients are allocated an ICorp containing one or more ISC and a VN, which connect them to the edges of the ASP's network. The client does not know how the ASP supports the resource allocation, where within the ASP's infrastructure it has been allocated resources, or even how much resource it has been allocated. Yet a client may still adjust the resource allocation as a function of its needs; the next section explains how.

2.3 Client-Delegated Control in the ICorpMaker

The coupling of server and network partitions into a single abstraction and creating an infrastructure that permits the creation of such entities, does not in itself solve the problem of how clients can specify their precise resource needs to the ASP. It is difficult for an ASP to refine something as vague as client satisfaction onto the concrete notion of buffer space allocated on switches, scheduling priorities on servers, etc., as it requires a very detailed knowledge of the client's activity and constraints.

Clients themselves are perfectly placed to know both their activity, their priorities and the required level of satisfaction. It makes sense therefore to delegate the activity of defining the resources the clients needs from the ASP to the clients themselves. Although the exact nature of the resources allocated to a client is hidden from it, a client knows when it does not have enough resources because the client's applications simply do not perform in a satisfactory way. Conversely a client can recognize that potentially it has a surplus of resources when its applications are idle.

Within the ICorpMaker, the ASP puts at the client disposal through a simple API, the means to increase or decrease the client's resource allocation. The decision as to how often and under what conditions to perform these operations is up to the client. This allows the client to take into account client-specific knowledge about the nature of the application in order to optimize resource allocation. For example, clients may expand their resource allocation if the service they are offering is more popular than they initially supposed and that revenue would be lost if the capacity were not increased.

Human clients can login into their physical server partitions, manually check the usage patterns and modify their resource allocation accordingly. More interestingly, as the infrastructure allows the safe execution of client application code within the context of an ISC, it can also safely run client *control* software. In order to make the process more dynamic, the necessary intelligence can be added to the software of the ISC itself. The ability to adjust the partition size is delegated to a dedicated separate process monitoring the set of tasks running on the same server partition; how often and under what circumstances it attempts to perform an adjustment is decided by the client. Commercial software, e.g. transaction processing, often maintain performance statistics. This information can be used by a monitoring task capable of taking advantage of the elastic nature of the environment. Figure 3 shows a schema of how control is delegated to clients within the ICorpMaker.

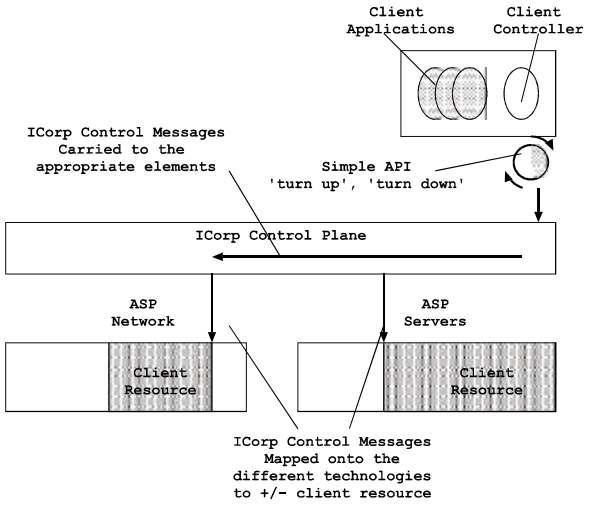

Figure 3 Client-delegated control.

Clients would expand their resource allocation to the maximum allowed one, unless there is some good reason for them not to. A pricing system is needed such that the cost to clients is some function of the resources they consume. Moreover, each time a client 'turns up' or 'turns down' the resource usage, they must be informed of the new charge. This paper does not consider pricing models, but assumes that reasonable models, encouraging the required behavior can be readily implemented within the `ICorpMaker`; [10] is one candidate charging model.

Clients who do not wish to dynamically change their `ICorp` need not, and may either simply manually perform the adjustments from time-to-time, or trust the ASP to do so. However, assuming such a system became widely accepted with a well defined interface, then one might expect software houses to write either dedicated software (for specific important clients) or general purpose software (that satisfy a large class of clients), which optimize their resource usage. Clients know which software is best as it is that which produces the smallest bill while performing the same service.

ASPs do not have to define complex service level agreements with clients nor offer the means for clients to observe that those agreements are being respected. Within the limits of the underlying physical structure clients can always grow their resource allocation until their applications behave satisfactorily. The only monitoring the clients perform is on their own applications whose expected behavior they should understand. In times of severe contention for resources, ASPs can use charging as a means of forcing their clients to back off.

This section has described how a flexible architecture in which the control of resource allocation can be delegated from ASPs to clients, would allow clients to take into advantage client-specific knowledge in order to optimize their resource usage. The next section describes how the `ICorpMaker` achieves this through a description of its architecture and implementation.

3 ICorpMaker ARCHITECTURE

3.1 Supporting ICorps over Multiple Technologies

Each configurable element within the `ICorpMaker` is managed by a software controller (`ICorpController`) running on or near the element. The set of controllers constitutes the `ICorp` control plane within which `ICorp` enabled elements exchange control messages, for example in order to initiate the creation of an `ICorp`.

`ICorp`s are supported over multiple different network and server technologies. Networks elements must distinguish between different `ICorp` traffics in order to forward it with the appropriate resource allocation to the appropriate servers. `ICorp` packets need to be labeled in some way in order that the network elements make the appropriate forwarding decisions. A packet label can be something

- implicit, e.g., an IP source address,
- explicit, e.g., an MPLS label [6],
- present only at the IP layer or also usable by layer-2 devices such as Ethernet and ATM,
- that is significant for the network as whole, e.g., a VLAN identifier, or
- has only local significance, e.g., an ATM VCI.

Explicit labels can be used exclusively for the purpose of identifying the `ICorp` to which traffic belongs, but require changing formats of well established data units (or creating new encapsulation for them) with all the inherent problems.

The `ICorpMaker` uses implicit labeling; no changes are required in the format of data units and no modification of the forwarding function of the nodes is required. The Internet protocol suite is sometimes described in terms of an hour glass shape: many different protocols exist above and below the IP layer, but are unified at the IP layer. Although in theory layer-2 protocols should be oblivious to IP addressing, in practice they are often IP-aware. For example an Ethernet switch may limit a broadcast to a VLAN containing only hosts within a given IP subnet. It achieves this by maintaining a MAC address/IP address mapping obtained by sniffing ARP requests and replies. Policy can be used to enhance the IP address/Physical Address resolution, for example, a given set of IP addresses are resolved to both an Ethernet MAC address *and* a specific VLAN priority tag.

Within the ASP's network the `ICorp` is supported by one or more dynamically created *private* IP

subnetworks [11], a private IP network address is one which is meaningful only within a given domain and is never exchanged across domain boundaries. Network Address Translators can perform the conversion from public to private IP addresses (and vice versa) at network boundaries. Each Instant Server Chunk (ISC) within the ICorp is allocated a dedicated private IP address. Network elements identify the packets of an ICorp based on their source and destination address, other additional information may be taken into account, e.g., IPv4 TOS bits, if the network element is capable of making use of it.

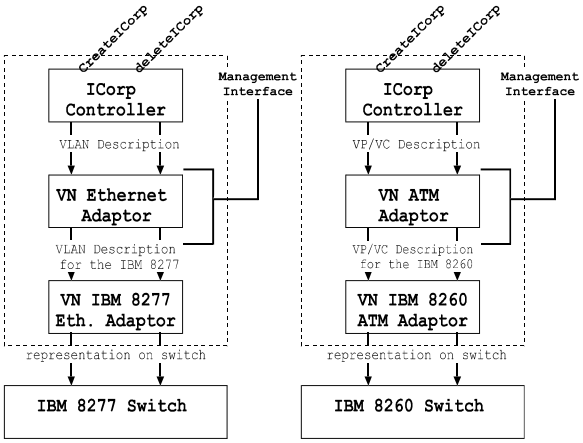

Figure 4 Example of adaptors.

During the creation of the private IP subnets the high level description of the network resources required by the client are mapped on to the appropriate underlying technology. Figure 4 shows an example of two adaptors for ATM and Ethernet switches. The initial allocation can be one chosen from a small set of predefined allocations, but, as Section 3.4 demonstrates, this can be refined by clients during the lifetime of the ICorp. Both layer-2 and layer-3 devices can offer support in regard to supporting network resource partitioning. For example an IP router might support diff-serv priority based queuing on bits in the IP header, an IEEE 802.1q [5] enabled Ethernet switch priority switching based on frame tags, and an ATM switch continuous bit rate [12] cell forwarding based on the virtual circuit identifier. In order to be as general as possible the infrastructure supporting the virtual networks should be able to handle different technologies.

The principle adopted is that the underlying node should try to do as *much as it can* to support the virtual network, but its exact behavior is a function of the technology and the precise capabilities of the element. This is called the *low-expectations principle*. It is only by adopting such a weak semantic that a wide variety of technology can be coordinated within the same framework. The ICorp control plane guarantees connectivity for the traffic of the ICorp, but gives no precise guarantee as to how the associated policies are enforced. The behavior of the ICorp control plane may be thought of as a type of 'best-effort policy support'. In some cases the element will not be able to do anything at all. For example, legacy IP routers do not support anything in the way of service differentiation—even in such cases, however, it is still useful to know the policy that the node is expected to support, regardless of whether it is enforceable.

3.2 Patterns of Communication in ICorp Creation

Clients communicate with a gatekeeper entity through a simple Web interface in order to specify their needs. The initial resource allocation is simply chosen from a small set of predefined values and clients can refine this over the lifetime of the ICorp. The gatekeeper uses the client description and its knowledge of the current state of the ASP's infrastructure to create an ICorpContract containing a description of the resources that the ICorp needs. Part of this description is the set of end-points—servers and access routers—that need to participate in the ICorp.

The protocol used for carrying ICorp control messages within the ASP infrastructure is RSVP [13] extended in ways to make it more adequate for the dynamic creation of ICorp. To avoid confusion, this variant of RSVP is termed RSVP-ICorp.

An RSVP-ICorp daemon is run as part of the ICorpController. In order to allow RSVP-ICorp messages to be propagated to both layer-2 network nodes and routers, the forwarding function of the daemon is a function of the nature of the device it controls. Routers use their routing tables in order to identify the next router to forward a message to reach a given host. They use physical topology information to determine which is the next layer-2 hop to take them to that router, the message is then forwarded to that layer-2 entity with the next router as subtarget.

The RSVP-ICorp daemon of layer-2 entities uses physical topology information to find the next hop to reach the *subtarget* rather than the end-point, e.g. an Ethernet switch determines the next Ethernet switch to cross to get to an IP router. The view of the physical network that an ICorpController has is restricted to the boundaries of the set of layer-2 technologies to which it is attached. For implementation purposes a simple Ethernet discovery system has been created based on each Ethernet ICorpController registering its port/MAC association with a central entity and that entity correlating the information to obtain the view of the entire Ethernet. This method could easily be replaced by appropriate layer-2 discovery mechanisms if available in the environment.

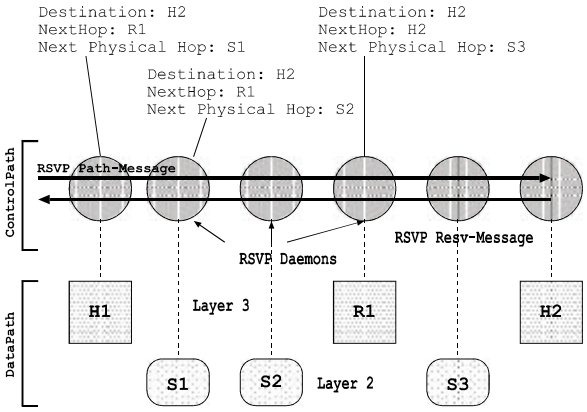

Figure 5 VN creation using RSVP messages.

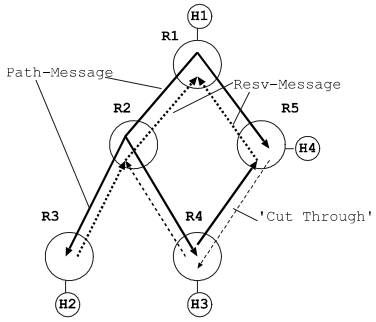

Figure 6 Optimizing the virtual network topology.

If a layer-2 network element is not discovered then it is transparent to the control path and no RSVP-`ICorp` message is forwarded to it. Data for the `ICorp` can still cross that network element, but no guarantee can be enforced on it. This is simply another application of the low-expectations principle mentioned in Section 3.1. Figure 5 shows the pattern of communication for creating a VN between two end-points `H1`, `H2`, interconnected at the IP layer by a router `R1` and at layer 2 across three switches `S1`, `S2` and `S3`. The `gatekeeper` chooses one of the end-points as the emitter of RSVP-`ICorp` path messages, these messages are propagated to all the others end-points. The `ICorpContract` is included in the RSVP-`ICorp` path message.

The RSVP-`ICorp` forward path messages reserve resources on the network elements and the reservation messages actually commit them in the network element. An `ICorpController` that receives an RSVP-`ICorp` path message, examines the contract to determine if it can support the contract and if so reserves the required resources. A given `ICorpController` can add state to the `ICorpContract`, which may or may not be meaningful to the next hop. For example, on an ATM switch the description of the created virtual connection is added so that the next hop can make use of it to create the entire virtual circuit.

If it is the `ICorpController` of a network node it then forwards the message to the next `ICorpController` upstream. If it is the `ICorpController` of a server specified in the contract and it is willing to support the `ICorp` it creates an `ISC` and it replies by sending RSVP-`ICorp` reservation messages upstream. An `ICorp-Controller` receiving a first reservation message for a given `ICorp` maps the resource reserved for that `ICorp` on the appropriate representation for that element. The ultimate receptor of all the RSVP-`ICorp` reservation messages coming from the other end-points in the `ICorp` is the `IS` Proxy resident on the initiating access router. This proxy is termed the leader proxy.

Implicitly a tree shaped `VN` is created with the initiator as the root of the `VN` and the others end-points as leaves. Although reachability is guaranteed, the route taken may be extremely inefficient, i.e., all communication between leaves must travel across their nearest common ancestor in the `VN` tree. The path message that a host receives contains the `ICorpContract`, therefore the receiving host has access to the complete set of other hosts participating in the virtual network. An end-point can make a local decision if there is a better route between itself and one or more of the other non-senders nodes and can start sending out path messages for the same contract. The `ICorpController` does not forward `ICorpContracts` over interfaces to which they are already supported; normal routing will ensure that the better route to the other hosts are included as part of the virtual network. Figure 6 shows a VN created using RSVP between four end-points `H1`, `H2`, `H3` and `H4`. In the initial phase, `H3` and `H4` have `R1` as nearest common ancestor. However, `H3` recognizes that there is a better route to `H4` and performs a ‘cut through’. In summary, a tree is created and then if better paths between the leaves exist, the leaves are joined in order to make a graph.

The `ICorpMaker` uses a soft state reservation model, the resource reservation for a client over a physical element must periodically be renewed or the resources on that element for that `ICorp` are released.

3.3 End-User Access to ICorp

At the end of a successful creation process, Instant Server Chunks (`ISC`s) running appropriate applications have been started on one or more of the physical servers and a virtual network has been created which interconnects these `ISC`s with the edge routers across which end-users will access these applications.

An IS Proxy creates the illusion that the set of server partitions is a single server. IS Proxies are started at access routers which receive RSVP-ICorp path messages. The IS Proxy knows from the contract the addresses of the ISCs and can perform TCP forwarding to them. IS Proxies by default perform a simple round-robin load balancing.

After the gatekeeper has successfully created the ICorp it dynamically updates one or more DNS servers, such that the DNS entry of the client ICorp resolves to the address of a router hosting an IS Proxy.

3.4 Client-Delegated Resource Allocation

An ISC not achieving the required performance because of insufficient resources can use information available locally to determine whether the congestion is local or remote, for example: the size of the TCP congestion window; the CPU occupancies of its process. Having identified a problem, the ISC will ask for an increase in resources within the reservation refresh message, giving some indication of the type of resources that need to be augmented, for example server memory, network bandwidth. The ICorpControllers on the elements across which this message passes will attempt to achieve the increase if possible. For example, the ICorpController on the server at which an ISC is asking for more CPU cycles, will attempt to allocate more processing power to that ISC. A reservation message carrying a request for more network capacity will be carried from the server to the edge routers and each of the network elements will attempt to readjust the capacity allocated to the corresponding VN. If the adjustment is insufficient then in the next iteration the ISC will send a reservation message asking for a further increase in the capacity.

The leader IS Proxy resident on the initiating access router periodically receives the RSVP-ICorp reservation messages sent by all the other end-points. It correlates this in order to make load-balancing decisions. The correlated information is then dispatched to all the other end-points, as part of the field of the RSVP path message. This allows other IS Proxies to adjust their behavior. The cost of an increased resource allocation to a client is also carried back to the ISC by sending an updated RSVP-ICorp path message. It is possible that an increase is not possible as the physical limits of the device have been reached. The leader IS Proxy recognizes this situation after having received the same reservation message requesting resource increase a certain number of times. If it is a request for increased server resources that cannot be met at the physical server, then the IS Proxy can request the gatekeeper for another candidate physical server and augment the ICorp to include another ISC at that server. If networks resources are the bottleneck, then the Proxy can attempt to reconfigure the network, for example by moving the location of the ISCs, adding new access points etc.

Section 3.1 described how the mapping between the abstract notion of a resource allocation and the actual underlying technology is achieved through an adaptor layer. The client's request for more of a resource is also dependent on the nature of the element and is also passed through the adaptor. The size of the increments and decrements that the adaptors use is set by management. Elements may not react to a user request for more resource either because: they are at full capacity all ready; the technology does not support partitioning or does not support modifying a partition dynamically; they do not control that resource, e.g., the reservation message carrying the request for more servers cycles is ignored by the ICorpController of an ATM switch, which simply forwards it on. If a client is continuously and unsuccessfully asking for more resources the leader IS Proxy will try to reconfigure the entire ICorp and notify a management system.

Reference [14] proposes a Network Element Control Protocol (NECP) to enable a physical server to indicate to a router that it is congested demonstrating the need for collaboration between servers and networks in order to optimize overall system performance. The framework described here extends an existing protocol—RSVP—, allows partitions of servers to both signal the need for more network and server resources and have the infrastructure take that demand into account, all within a single coherent system. The entity best capable of identifying a congestion problem, the application, is also that which initiates the allocation of more resources. How the application identifies a problem and reacts to it, is up to the application itself, existing mechanisms such as TCP congestion control fit naturally into the scheme, e.g. after a certain number of ACKs have been long lost, increase the capacity of the VN. Clients do not need to know precise details about how resources are allocated, only that they need more, or can do with less.

4 ICorp IMPLEMENTATION

This section complements the description of the ICorpMaker given in previous sections, by outlining how ICorps are created in our test network.

4.1 Implementation of the Virtual Network

The test network is arranged such that there are two Ethernet islands interconnected via ATM. Each Ethernet island contains a small number of 100 MB/s

Ethernet switches (e.g., IBM8277, IBM8371) and are connected to the ATM network via AIX workstations acting as routers. Attached to one of the Ethernet islands are a collection of servers running Linux and Windows NT, whereas attached to other are a set of access routers (Cisco 3600's) through which clients on the public network access those servers.

While the concept of a VLAN is available on most Ethernet switches, their precise realization are vendor and even switch specific. Therefore coordinating the action of VLANs across different vendors switches is problematic. However, most implementations support the concept of an IP-based VLAN, i.e. one in which the traffic associated with a given IP subnet is automatically associated with a given VLAN. Using IP addressing as the means of labeling `ICorp` traffic and IP-based VLANs as the means of differentiating between them permits different vendor VLAN implementations to have a common understanding about how traffic for a given `ICorp` should be treated. The `ICorp` control plane allows the creation of the VLAN to be coordinated in this heterogeneous environment without requiring switches to speak each others' proprietary VLAN control protocols. For switches that support IEEE 802.1q the VLAN is given an appropriate priority, allowing traffic belonging to high priority `ICorp`s to be privileged over lower ones at those switches.

On the AIX 4.3 router, the routing tables are updated such that the routers forward packets on the new private IP subnets. The routes to the new private subnets must be via the same physical interfaces as the corresponding public ones as they are simply an overlay. A set of permanent virtual circuits (PVCs) each with a different traffic class interconnects the two AIX boxes across the ATM network. The router chooses an appropriate PVC for the demanded service level, in order to forward traffic to the remote router. No controller is run for our ATM switches as this is part of the production network and experimental systems are not allowed to change the state of production switches.

Typically a client's VN within our testbed will be supported as two private IP subnets each of which is supported by a single VLAN. One of the VLAN interconnects all the appropriate servers with the AIX router linked to the ATM network. The other VLAN connects the homologue AIX router to all the access routers. The two AIX routers are updated such that traffic exchanged between these two private IP subnets is carried over an appropriate ATM PVC. A client's request for more network resources gets translated into an attempt to change the PVC over which their traffic is carried to one with a larger capacity.

The intention is to use pricing to force clients to back-off when requested capacity starts approaching actual capacity; the client's `ICorp` time interval cost being carried within the RSVP-`ICorp` Path-Message. Note that whereas clients can always ask for more resources, the infrastructure does not guarantee that they will get them. Initially the infrastructure tries to increase the amount of capacity of the `ICorp` without changing its shape. If the capacity cannot be increased in the appropriate way on the actual `ICorp` and the client's requests continue, then the system attempts to change the shape of the `ICorp` to satisfy the demand. The time scales on which RSVP-`ICorp` refresh messages (and consequently resource modification requests) are sent and acted on are of great importance to the performance of the system. On-going work is trying to obtain a best understanding of the issues through detailed performance analyses.

4.2 Implementation of the Instant Server Chunks

Within the current implementation, `ISC`s are implemented using commercial software, VMWare [9], which allows the execution of one operating system within another. This resident operating system is called the host OS, whreas those that execute within the VMWare framework are called guest OSs. Guest OSs have their own IP and MAC layer addresses, the host OS acting as a bridge on their behalf. An `ISC` corresponds to a single guest Linux OS, started when a request for an `ICorp` creation is received. The Linux boot process is modified such that an `ISC` controller daemon is executed before the login shell is started. This daemon starts the necessary binaries on behalf of the client, either well known within the OS, e.g., *ftp* server, or client supplied (identified by the URL where the binary is to be obtained). Client binaries may cause their own guest OS to fail, but cannot hinder those of other clients or the host OS itself. The guest OS appears as a process within the host OS and within a Unix system can be allocated a priority. The guest OS's hard disk, is a file resident within the host OS whose size can be set by the host OS. The guest OS by default is set to a low priority with a small disk. The `ISC` running on it sends out reservation messages to the `ICorpController` on the host OS. A simple monitor program checks the resource usage of the guest OS, if CPU usage exceeds a certain threshold for a period of time, then the `ICorpController` can increase the priority of the guest OS process.

Figure 7 shows the sequence of events at a server in order to instantiate an ISC. **(1)** an RSVP-`ICorp` path message is received from the `ICorpController` at the egress network element. The `ICorpController` uses the contract in the message to create the context in which VMWare is executed. **(2)** The VMWare implemented ISC is started; the last step of the guest OS boot process is the starting of an ISC controller, this controller starts

the client applications, and replies to the `ICorpController` with an RSVP-`ICorp` reservation message (**4)**. The `ICorpController` then returns the reservation message to the controller on the connected network element **(4)**.

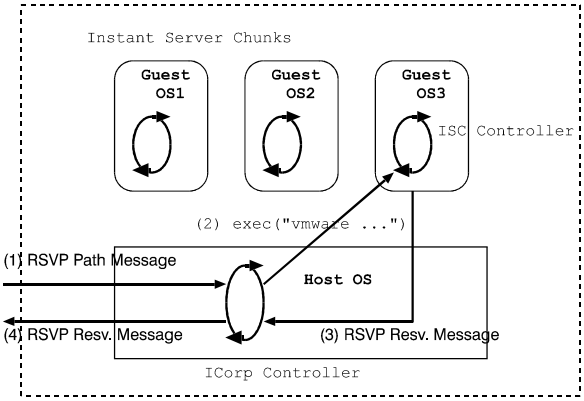

Figure 7 Implementation of VN.

Current work is looking at using other partitioning mechanisms, for example [7], to support `ISC`s.

5 CONCLUSION

Defining the amount of network and server resources that a client requires from an ASP to perform a given activity is problematic: in general, clients cannot be expected to know details about the ASP's infrastructure, nor ASPs the nature of the client's activity. A dynamic framework has been presented which using suitable abstractions and protocols allows the task of resource allocation to be delegated from the ASP to the client themselves. Clients may request an increase or decrease in the size of the resource allocated as a function of the desired application behavior and the amount of money they are prepared to spend, but without requiring a detailed knowledge of the underlying technology. Clients do not have to be able to define their precise requirements a priori and the framework allows clients to vary their resource allocation dynamically over time as circumstances change.

REFERENCES

[1] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Service," Internet RFC 2475, May 1998.

[2] B. Glesson, J. Heinanen, G. Armitage, and A. Malis, "A Framework for IP Based Virtual Private Networks,"' February 2000.

[3] S. Fotedar, M. Gerla, P. Crocetti, and L. Fratta, "ATM Virtual Private Networks," *Communications of the ACM* **38**, pp.101-109, 1995.

[4] D. Jamieson, B. Jamoussi, G. Wright, and P. Beaubien, "MPLS VPN Architecture," draft-jamieson-mpls-vpn-00.txt, August 1998. Work in progress.

[5] IEEE/ISO/IEC, "Virtual Bridged Local Area Networks," *ISO Publication*, July 1998. Draft Standard: IEEE Standard for Local and Metropolitian Area Networks, P802.1Q/D11.

[6] E. Rosen, A. Viswanathan, and R. Callon, "A Proposed Architecture for MPLS," draft-ietf-mpls-arch-06.txt, August 1999.

[7] Ensim, "ServerXChange, A Complete Service Deployment Platform," Technical White Paper, October 1999.

[8] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford, "Xenoserver; Accounted execution of untrusted code," in Proceedings of Hot Topics in Operating Systems, 1999.

[9] VMWare, "Getting Started Guide, VMWare 2.0 for Linux," VMWare Technical Support, January 2000.

[10] P. Key, D. McAuley, and P. Barham, "Congestion Pricing for Congestion Avoidance," Microsoft Res. Techn. Rep. MSR-TR-99-15, Feb. 1999.

[11] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918, February 1996.

[12] S.S. Sathaye, "ATM Forum Traffic Management Specification Version 4.0," in ATM Forum Technical Committee, Contribution 95-0013, 1995.

[13] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource ReSerVation Protocol," *IEEE Network* **7**, pp. 8-18, September 1993.

[14] A. Cerpa *et al.*, "NECP, the Network Element Control Protocol," Internet Draft, February 2000.