A Dynamic Pricing Framework to Support a Scalable, Usage-based Charging Model for Packet-switched Networks

Mike Rizzo, Bob Briscoe, Jérôme Tassel, and Konstantinos Damianakis

Distributed Systems Group, BT Labs, Martlesham Heath, Ipswich IP5 3RE, England
{michael.rizzo,bob.briscoe,jerome.tassel,konstantinos.damianakis}@bt.com

Abstract. We describe a dynamic pricing framework designed to support a radical approach to usage-based charging for packet-switched networks. This approach addresses various scalability issues by shifting responsibility for accounting and billing to customer systems. The ultimate aim is to create an active multi-service network which uses pricing to manage supply and demand of resources.
In this context, the role of the dynamic pricing framework is to enable a provider to establish 'active tariffs' and communicate them to customer systems. These tariffs take the form of mobile code for maximum flexibility, and the framework uses an auditing process to provide a level of protection against incorrect execution of this code on customer systems. In contrast to many active networks proposals, the processing load is moved away from routers to the edge of the network.

1 Introduction

As the Internet continues to grow and evolve into a global, multi-service network, the issue of how to charge fairly and sensibly for network services is becoming increasingly relevant. The flat-rate charging model, currently used by virtually all ISPs worldwide, relies heavily on characteristics of the present best-effort Internet which may no longer be valid in the near future. For example, the inability of the present Internet to offer differential services means that it does not make sense to speak of higher prices for better services. And the bandwidth limitations associated with dial-up connections provide a convenient cap on resource usage by any one individual, thereby protecting providers' routers from being hogged by a single user at the expense of other users.

It is envisaged that the current best-effort Internet will gradually be replaced by a network that can offer differential levels of network service that are better suited to the individual needs of specific applications. For example, a video-on-demand application might make use of a high-bandwidth, low-jitter, reservation-based service, whilst email would continue to use best-effort service. Furthermore, it is expected that access bandwidth for end-users will increase in order to enable the provision of high quality multimedia services. In this scenario, the flat-rate charging model gives rise to the anomalous situation wherein users that make heavy demands on network resources are charged the same amount as users that make lighter demands. Moreover, the service offered to lighter demand users is likely to be impaired by the provision of services to higher demand users. This situation is likely to result in either convergence back towards a single-service network (where users will always request the best service possible), or denial of service to users whenever network resources are operating at maximum capacity.

It makes sense, therefore, to abandon flat-rate charging in favour of a usage-based model in which there is a relationship between price and resource usage. Indeed it is possible that such a model might also have been considered for the current best-effort Internet, were it not for the substantial increase in operational complexity involved. Whilst flat-rate charging is extremely easy to implement, usage-based charging requires that some form of usage accounting be carried out before a charge can be computed. It is generally accepted that the additional operational cost associated with such accounting is substantial due to the increase in processing power that is required, not only to cope with accounting processes as such, but also to compensate for the the blocking nature of the measurement process, which has a negative effect on throughput. Consequently usage-based models are not yet considered viable, and many proposals have focused on compromise solutions based on aggregation [3, 9, 8, 11].

As part of a project investigating radical approaches for operational support systems, we are investigating the possibility of lowering the operational cost of usage-based charging by shifting responsibility for billing to the users themselves. We propose that users measure their own traffic, and compute their own bills using tariffs supplied by the provider. This spreads the load so that each processing unit uses a near-negligible amount of resources for billing purposes, all the more so when one considers that most users' machines spend much of their time in an idle state. It also allows network routers to focus on their principal function without sacrificing throughput.

Using this approach it is possible to charge users differentially on the basis of both volume and quality of services. This gives providers a degree of control over resource usage, because tariff structures can be determined so as to give users an incentive to use the minimal amount of resources that meets their requirements. Furthermore, finer-grain control over supply and demand management of network resources can be achieved by price variation along the lines of established economic supply and demand principles. At times when resources are in short supply, demand is curbed by raising service prices. Conversely, at times when resources are under-utilised, demand is stimulated by lowering prices. If the decision-making involved in changing prices is (partially) automated[1], then the result is an intelligent active network which performs its own supply and demand management.

[1] In this paper we limit ourselves to describing a general framework which can support this concept. It is beyond the scope of the paper to present arguments related to the desirability or extent of automated decision-making in this regard.

This approach immediately raises several questions, particularly with respect to trust, stability, security, and user acceptance. Some of these questions are briefly covered in sections 2 and 3, and are expanded on in another paper [6]. The principal focus of this paper, however, is the framework which enables a provider to communicate tariffs and price variations to its customers. Following a broad overview of our approach to charging in Section 2, Section 3 outlines the issues of specific concern to tariff representation, dissemination, and application. Section 4 describes a prototype that was developed to demonstrate the approach, and to gain experience in the choice of suitable implementation techniques. Section 5 follows with indications for further work. Finally Section 6 concludes with some general implications for active networks.

2 Background

We assume a packet-switched network in which a variety of network services are made available to users. The exact nature of these services, and the specific characteristics that form the basis upon which they might be differentiated, are not important for our purposes, and may vary from one provider to another. However we assume that, in general, service usage may be measured on the basis of packet counts, and may be classified using some notion of quality of service, irrespective of whether this is reservation-based [4] or class-based [2].

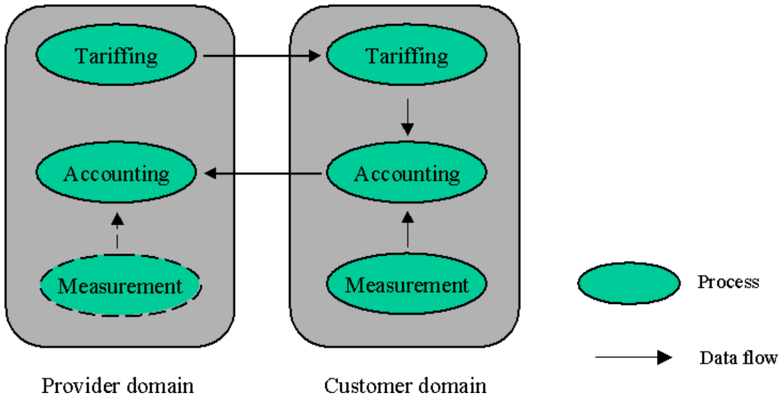

Fig. 1. Radical charging model: processes and flow of information

In the proposed charging model, the customer system is responsible for accounting for usage under the instruction of the provider. The provider supplies tariffs for each of the available services, along with other information pertaining to their application e.g. how frequently they should be applied. The customer's system measures and categorizes both inbound and outbound traffic, applies the appropriate tariffs for each category of traffic, and periodically sends accounting reports to the provider. The customer's system might also be responsible for making payments, although this may be delegated to some other entity. The various processes and data flows are depicted in Fig. 1.

This model clearly places a lot of trust in customer systems. Our view is that this does not pose a problem, as long as the provider is able to check up on a sample of its customers from time to time. A random audit function may be employed by the provider's accounting process to make measurements pertaining to a particular customer at the provider end, and verify that the customer's accounting reports tally with the observations made.

The model is not targeted solely at the edge of a packet network, but is intended to be applied recursively throughout the network. Thus an access provider might be the customer of a larger provider, which may in turn be the customer of a backbone provider. A multi-host edge customer might also employ a similar model within its network in order to recover costs.

Whilst it is likely that charge for network use will be uni-directional at the edge of the network, this is not the case in general. The distinction between provider and customer becomes somewhat blurred as one approaches the core of the network. There is also a charging issue related to the direction of traffic: should a chargeable entity pay for packets sent, packets received, or both? In general there are four possible charges between two entities A and B:

- A charges B to send packets to it;
- A charges B to receive packets from it;
- B charges A to send packets to it;
- B charges A to receive packets from it.

An entity, therefore, can assume the roles of both provider and customer with respect to some other entity. We allow charging for any combination of the above to allow maximum flexibility with respect to traffic direction when establishing charging policies and tariffs.

3 The Tariffing Subsystem

Having outlined the general principles of our charging model, this section focuses on the role of the tariffing subsystem, which comprises:

- establishment and adjustment of tariffs on the provider side;
- dissemination of tariffs and adjustments to customer systems;
- application of tariffs by customer systems to local measurements.

The remainder of this section characterizes the requirements attached to this role, setting the scene for the subsequent section.

3.1 The Nature of Tariffs

We assume that a provider may set a separate tariff for each category of service that it offers, and that for a particular category of service, a tariff may change periodically. We do not exclude the possibility that a tariff may change frequently e.g. in response to changing traffic patterns. However, we distinguish between two kinds of change, namely *replacing* a tariff and *adjusting* a tariff. The former implies substitution of an old tariff by a new one, whilst the latter involves 'tuning' an existing tariff. We envisage that tariff adjustments will occur more frequently than tariff replacements.

There is a clear distinction between 'tariff' and 'price'. A tariff is responsible for determining a price with respect to a set of given contextual parameters. It is therefore possible for a price to change without there being a change to the tariff that determines it. For example, a traditional PSTN tariff might offer one price for peak hours, and another price for off-peak hours. Here price varies according to the time of day, but the tariff remains constant.

Continuing with this example, a tariff adjustment might involve changing the off-peak price, or perhaps the times of day at which off-peak is considered to start. If an altogether different tariff structure is required e.g. due to the introduction of a new discount scheme, then a tariff replacement is required.

It is a goal of the model to allow maximum flexibility in the structure of tariffs. Ideally it should be possible for tariffs to be modelled on complex rules. For example, a provider may wish to deploy a tariff for best-effort traffic which operates such that customers are penalized if their systems do not back off in the presence of congestion. It is also desirable to put as much intelligence as possible into tariffs, so as to avoid frequent transmission of tariff changes to customers. This is particularly relevant at times when network congestion is high, in which case a tariff should be capable of making price adjustments without having to receive explicit instruction from the provider.

3.2 Supply and Demand Management

Our model allows on-the-fly changes to prices, and can be used to support to supply and demand management wherein prices fluctuate on the basis of current demand. This concept may come across as too radical to some. However it is worth pointing out that many people are quite happy to purchase variable rate mortgages, or invest in the stock market. And just as other people pay a fee for a fixed-rate mortgage, or are prepared to commit themselves to a safer long-term savings plan, it is quite conceivable that they will be prepared to pay for their price to be kept fixed, or for price variations to be constrained in accordance with some pre-defined contract. The notion that there may be different charging schemes for a given service category leads us to the concept of a *product*. For example, a product A might offer best-effort service at a fixed price, whilst another product B might offer best-effort service at a variable price.

It is envisaged that a provider will adjust product prices on the basis of observations it makes with respect to:

- the prices it is being offered by its own providers;
- competitors' prices;
- current resource utilisation;

- relative demand for different products e.g. the price for a particular product might be lowered so as to entice users to switch to it.

Price adjustments can be effected in one of three ways:

- A tariff may be able to adjust prices on the basis of observations made by local monitoring, without necessitating explicit communication from the provider. This requires foresight at the time the tariff is designed, and is limited to those price variations which are dependent exclusively on observations local to the customer system.
- The provider may tune a tariff by adjusting some of its parameters. This kind of adjustment is required when the decision is dependent on observations which cannot be made by customer systems e.g. variations in the prices offered to the provider by its own providers, and the changes required can still be accomodated by the present tariff.
- The provider may replace a tariff. This is required when the present tariff cannot accomodate the changes that are required.

The first of these is by definition an automated decision. The second may be performed both manually or by an agent that issues adjustments on the basis of observations made by the provider system. The third is likely to be performed manually, as replacement of a new tariff represents a major change in business strategy. In particular, creation of a new tariff involves an element of design which can only be sensibly carried out by a human with expertise in economics. However, it is possible that given the availability of a repertoire of tariffs, an agent might be employed to automatically switch tariffs for a product on the basis of a set of specified rules.

Given the possibility of frequent, on-the-fly changes, it is important that customers have some way of knowing what is going on. It is difficult to construct a customer user interface that can convey the workings of a tariff if the tariff is not known at the time the customer software is deployed. It is therefore desirable that the rules that define tariffs are accompanied by user interfacing suggestions that could somehow be used by the customer system [2].

4 Implementation

This section describes a prototype that we implemented to demonstrate the tariff subsystem outlined above. The key features of our design include:

- using mobile code to represent tariffs and associated graphical user interface (GUI) components;
- use of a repeated multicast announcement protocol to communicate tariffs and tariff adjustments efficiently;

[2] This is intended for feedback purposes only. We envisage that the customer system will also employ an agent (potentially supplied by a regulator) to monitor tariffs for the purposes of verifying that they are within the bounds of the contract.

– using dynamic class loading and reflection in order to receive and tune tariffs.

The prototype comprises two applications, namely:

– a provider system which allows the provider to introduce, replace, and tune tariffs for a number of products;
– a customer system that enables customer to keep track of the charges being applied for the products they are using.

The provider system is intended to serve multiple instances of the customer system running on different hosts in a multicast-enabled network. A multicast protocol is used to communicate tariff data to customer systems.

4.1 Tariff Representation

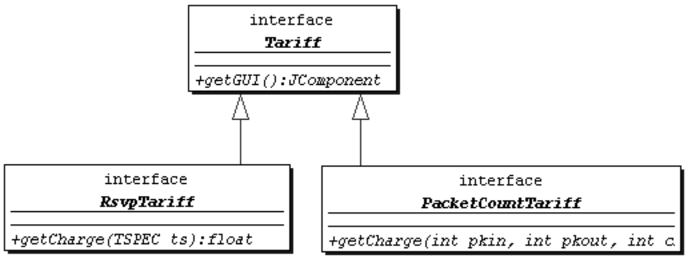

Fig. 2. UML description of tariff definition framework

In order to maximize flexibility with respect to the definition of tariffs, we chose to represent tariffs using Java classes. This technique also proved useful for supplying custom-built GUI components to support visualisation of tariffs.

Figure 2 illustrates the framework within which tariffs are defined. The `Tariff` interface acts as the base class for all tariffs. This defines a single operation `getGUI()` which returns a Java SWING component that can be incorporated into the customer's GUI. The intention is that this GUI component will enable the customer to visualise the behaviour of the tariff using the most appropriate user interfacing techniques for that tariff. Interfaces derived from `Tariff` establish a set of tariff types, each of which is associated with a different set of measurement parameters. These parameters are identified by listing them in the signature of the `getCharge()` method. For example, the interface `RSVPTariff` defines `getCharge()` as receiving an RSVP `TSPEC`, allowing for the definition of tariffs that compute price on the basis of the characteristics of an RSVP reservation [1]. Another interface, `PacketCountTariff`, defines `getCharge()` as receiving measurements of packets in, packets out, and current congestion (typically measured as a function of packet drop), allowing for the definition of tariffs that are dependent on packet counts and sensitive to congestion.

Tariffs are defined by providing implementations of tariff interfaces. For example, `PacketCountLinear` implements `PacketCountTariff` to compute charges in proportion to packet counts. `CongestionSensitiveLinear` works on a similar basis, but adds a penalty charge if the customer does not stay within specified traffic limits in the presence of congestion.

A tariff implementation may make use of other 'helper' classes to assist it in its operation, as well as one or more GUI component classes for customer visualisation purposes. A GUI may also be required to enable the provider to make tariff adjustments. A complete tariff description then, consists of a set of Java classes, some of which are destined for the customer system and others which are intended for use by the provider system. The customer-side classes are bundled into a Java JAR file to facilitate loading by the provider system.

4.2 Tariff Dissemination and Adjustment

In order to deploy a new tariff, the provider system first loads the tariff classes which it requires into its execution environment. It then loads the customer-side bundle, serializes it, signs it with a private key (to enable authentication by customers), and uses an announcement protocol to distribute it to customer systems. Upon receiving the bundle, each customer system verifies the signature, unpacks the bundle, and loads the classes into its execution environment using a purpose-built dynamic class loader. An instance of the received tariff class is created and installed in place of the previous tariff. If the tariff has a GUI component (obtained by calling the tariff object's `getGUI()` method), then it replaces the GUI of the previous tariff. The change in GUI serves to notify the user that the tariff has changed.

Tariff adjustment involves the remote invocation of an operation which is specific to the tariff currently in force. This means that a customer system cannot know the signature of this operation in advance of receiving the tariff i.e. the operation will not be listed in any of the tariff interfaces known to the customer system. In order to get around this problem, use is made of the reflection feature supported by Java. In order to disseminate a tariff adjustment, the provider creates an instance of an `Invocation` object, which stores the name of the operation to be called, together with the parameters that are to be supplied to it. This object is then serialized, signed, and announced using the announcement protocol. When an adjustment is received and verified by a customer system, the `Invocation` object is de-serialized and applied to the current tariff by using reflection to invoke the described operation.

In order to simplify the announcement protocol, adjustments are required to be idempotent and complete. Idempotency guarantees that a tariff will not be adversely affected if an adjustment is applied more than once. Completeness implies that an adjustment determines the entire parameter set of a tariff object, so that an adjustment completely removes the effect of any previous adjustments.

4.3 Tariff Application

The customer system applies a tariff by repeatedly invoking the `getCharge()` operation supported by that tariff every second, and adding the returned value to the cumulative charge. The parameters supplied to `getCharge()` depend on the kind of tariff currently in force. For example, if the tariff is an implementation of `PacketCountTariff`, then measurements of inbound packets, outbound packets and congestion over the past second are required. However, if the tariff is an implementation of `RsvpTariff`, then only a `TSPEC` describing the current reservation is required[3]. Each invocation of `getCharge()` also results in an update to the tariff-specific GUI e.g. in `CongestionSensitiveLinear`, the usage parameters supplied to `getCharge()` are used to update the graphical displays of traffic and congestion.

4.4 Announcement Protocol

The announcement protocol is used to communicate serialized tariffs and adjustments from a provider system to multiple customer systems. The number of customer systems is assumed to be large, and a repeated multicast solution in the vein of SAP [10] is adopted.

Each product supported by a provider is assigned a multicast channel for announcement purposes. Customer systems listen to the channels corresponding to the products that they are using. For each product channel, the provider repeatedly announces the current tariff and the most recent adjustment made to it (if any). Each announcement carries a version number, which is incremented each time the announcement is changed. Customer systems only process announcements when a version number change is detected. If a new customer joins a channel, it waits until it receives a tariff before processing any adjustment announcements. Furthermore, an adjustment is only applied if its announcement version is greater than that of the current tariff, thereby ensuring that a missed tariff announcement does not result in the application of a subsequent adjustment to an old tariff.

4.5 Illustration

Figure 3 shows the GUI for the customer-system with the GUI component for the `CongestionSensitiveLinear` tariff embedded within it. The latter displays information about traffic and congestion levels, and indicates traffic limits which must be observed when congestion is above a specified threshold. The formula used to compute the current price is displayed in the bottom right corner. In the case depicted, congestion is above the threshold and the incoming traffic level is above its limit, with the result that a penalty of 0.2 is added to the price.

The upper part of the customer GUI displays the current charge being applied (per second), and the total charge accumulated by the customer. The GUI also

[3] Mention of this tariff is intended purely for illustration purposes, and does not necessarily represent a realistic or sensible way to charge for RSVP reservations.

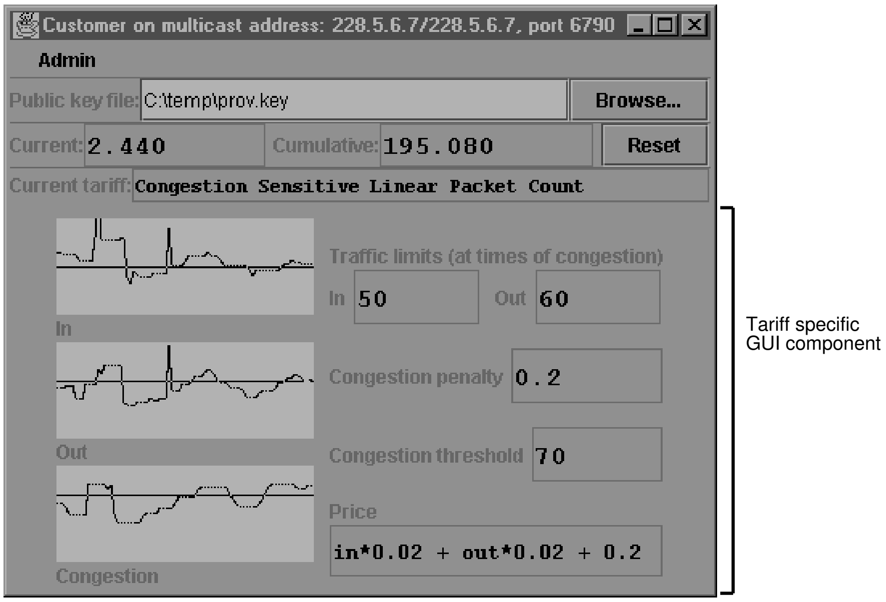

Fig. 3. Customer interface

allows the user to specify the public key to be used for authentication purposes, and shows details of the multicast address being listened to for announcements.

The provider system GUI consists of a set of product windows, each allowing control over a particular product. Figure 4 shows the provider-side window corresponding to the product being used by the customer system shown earlier. The lower part of the window contains the provider-side GUI component for the `CongestionSensitiveLinear` tariff. Using this interface, the provider can manually adjust the parameters associated with the current tariff. Any adjustments are communicated to customer systems using the announcement protocol, and are immediately reflected in customer-side GUI components.

The upper part of the product window allows the provider to replace the tariff currently associated with that product. Each tariff is fully described by a policy, which contains such details as a tariff name, a descriptive string, and more importantly the location of the JAR file on the provider's file system. Once a new policy has been selected, the 'Activate' button injects the corresponding tariff into the network, instantly replacing the existing tariff for that product.

5 Further Work

To date we have focused primarily on the establishment of general principles related to dynamic pricing, and the technical infrastructure required to support these principles. However, we have not identified those specific configurations of

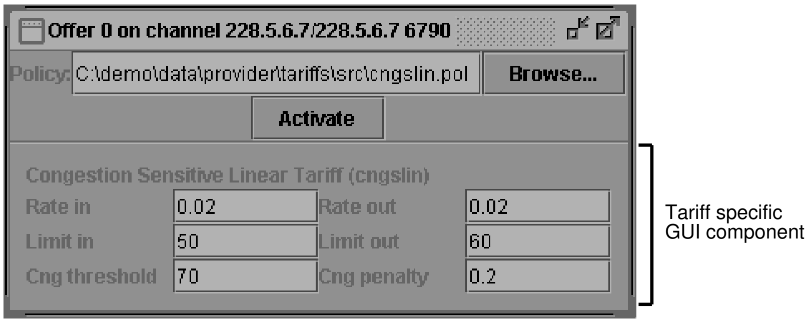

Fig. 4. Product window (in provider GUI)

the framework which are economically viable or socially acceptable. This can only be achieved by a combination of rigorous modelling, and experimentation with user trials. We intend to continue to develop the existing infrastructure into a testbed for experimenting with different tariffing schemes and for conducting user trials in order to gain experience with relevant human factors issues.

We are currently working on improving on a number of aspects of our current implementation, particularly with respect to the announcement protocol. Currently this makes a number of assumptions which are not valid in general. For example, it assumes that a tariff will fit in a single datagram. At best this leads to a packet fragmentation problem, but at worst it means that larger tariffs cannot be announced. There is also a problem in that well-known announcement addresses are expected to be known in advance by customers. This does not give the provider any flexibility with respect to channel assignments e.g. the provider may wish to move a product to share a channel with another product if it observes that a large number of customers are using both products simultaneously. Last but not least, there are a host of timing issues which need to be addressed e.g. working with multiple independent physical clocks.

6 Concluding Remarks

The dynamic pricing framework described in this paper demonstrates an active network approach to demand and supply management of network resources. This is relevant to the debate over whether overprovisioning is likely to be more cost-effective than rationing of resources in a multi-service network [12, 5]. By lowering the operational cost of usage-based charging, and by providing an infrastructure within which resource rationing mechanisms can be adjusted and fine-tuned as required, many of the arguments against resource rationing are invalidated.

Additionally, our experience with the dynamic pricing framework has some interesting bearings on the general areas of active networks and mobile code. One important point relates to the fact that active networks need not rely solely on the processing capacity of provider equipment. In particular, for applications with high processing loads, it may be possible to shift much of this load right up to the very edge of network, using multicast technology for efficient deployment of mobile code. Furthermore, it may be possible to exercise some control over core network elements as a side-effect of mobile code deployed to the edge of the network. In this respect, the dynamic pricing framework provides an interesting contrast to mainstream thinking on active networks, where the emphasis is normally on deploying mobile code to network routers.

Another point relates to the well-known security problem concerning the protection of mobile code from malicious or erroneous execution platforms. The sample-based auditing approach adopted in our charging model does not represent a complete solution to this problem, but is a reasonable compromise which can detect some cases of abuse whilst acting as a deterrent in general.

The class loader used for deployment of mobile code in our implementation differs substantially from other approaches to dynamic loading of remote classes in Java, as exemplified by Bursell *et al*[7]. Instead of loading each class individually from a remote class repository using a request-reply 'pull' protocol, we employ a 'push' approach in which a bundle of classes is delivered to the receiver in a single transaction. The receiver can then load all the classes without having to access the network. This is useful in situations where the sender determines which classes the receivers should be loading, and has the advantages that the effects of network latency are minimized, and that multicast may be employed to push bundles to several receivers simultaneously.

References

1. S. Berson, R. Lindell, and R. Braden. An architecture for advance reservations in the internet. Technical report, USC Information Sciences Institute, July 1998.
2. S. Blake et al. An architecture for differentiated services. Request for Comments (Proposed Standard) 2475, Internet Engineering Task Force, December 1998.
3. Roger Bohn et al. Mitigating the coming internet crunch: multiple service levels via precedence. Technical report, University of California, San Diego, Nov 1993.
4. R. Braden et al. Integrated services in the internet architecture: an overview. Request for Comments (Proposed Standard) 1633, IETF, Jun 1994.
5. Lee Breslau and Scott Shenker. Best-effort versus reservations: A simple comparative analysis. In *Proceedings of SIGCOMM '98, Vancouver*, 1998.
6. Bob Briscoe et al. Lightweight, end to end usage-based charging for packet networks, 1999. `http://www.labs.bt.com/projects/mware/charging.htm`.
7. M.H. Bursell et al. A mobile object workbench. In *Mobile Agents '98*, 1998.
8. David D. Clark. A model for cost allocation and pricing in the internet. In *MIT Workshop on Internet Economics*, March 1995.
9. Jon Crowcroft. Pricing the internet. In *IEE Colloquium on Charging for ATM (ref. no. 96/222)*, pages pp1/1–4, November 1996.
10. M. Handley. SAP: Session announcement protocol. IETF Draft, Nov 1996.
11. Frank P. Kelly. *Charging and Accounting for Bursty Connections*, pages 253–278. MIT Press, 1997.
12. Andrew Odlyzko. The economics of the internet: Utility, utilization, pricing, and quality of service. In *Proceedings of SIGCOMM '98, Vancouver*, 1998.