Market Managed Multi-service Internet

M3I

European Fifth Framework Project IST-1999-11429

Deliverable 2

Architecture Part I

Principles

The M3I Consortium

Hewlett Packard Ltd, Bristol UK (Coordinator)
Athens University of Economics and Business, GR
BTexact Research, Ipswich GB
Eidgenössische Technische Hochschule, Zürich CH
Technical University of Darmstadt, DE
Telenor, Oslo NO
Forschungszentrum Telekommunikation Wien Betriebs-GmbH, AT

For more information on this document or the M3I project, please contact

Hewlett Packard Ltd
European Projects Office
Filton Road
Stoke Gifford
BRISTOL BS34 8QZ
UK
Phone: (+44) 117-312-8631
Fax: (+44) 117-312-9285
E-mail: sandy_johnstone@hp.com

Document Control

Title	Architecture Part I: Principles
Type	Private deliverable (planned for eventual public dissemination)
Authors	Bob Briscoe
email	<mailto:bob.briscoe@bt.com>
Origin	BT Research
Wk package	3
Doc ID	arch_pt1_m3i.pdf

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V1.0	07 Jul 2000	Bob Briscoe	First Issue
V1.1	25 Oct 2001	Bob Briscoe	Formatting changes & minor edits
V1.2	28 Oct 2001	Bob Briscoe	Structure for split into 2 parts
V1.3	27 Nov 2001	Bob Briscoe	Far more depth on motivation behind principles in completely new section on fundamentals.
V1.4	11 Feb 2002	Bob Briscoe	Altered section 2 based on review comments. Finished all other sections
V2.0	21 Mar 2002	Bob Briscoe	Improved introduction; clarified applications overview and fixed citations of recent M3I deliverables
V2.1	27 Aug 2003	Bob Briscoe	Removed confidentiality markings

Contents

1 Introduction

We present an architecture for the Internet that both enables and manages network supply and demand using market mechanisms. The *technical* approach a network takes to controlling its load is typically termed its 'control architecture'. This is defined by the protocols and algorithms used and determines how tightly customers can control quality of service (QoS). On the other hand, the provider's *commercial* approach is defined by the contractual offer it makes, in particular the tariff on offer. To avoid using the term 'architecture' for two things, we use the term **service plan** for this combination of a provider's technical and commercial approaches. The M3I Architecture encompasses the internal working of each service plan as well as the overall approach to combining all service plans across the Internet. For instance, the Differentiated Services Architecture (Diffserv [24, 9]) is 'just' a service plan by our definition, as it defines not only its QoS signalling technologies, but also service level agreements, thus defining the form of contract the customer is expected to agree to, how the value chain is arranged, etc. As well as existing service plans like Diffserv, the M3I Architecture enables a rich variety of novel service plans in different networks. That is, specific technical control architectures and specific commercial approaches interworking across an internetwork. Network providers are then able to differentiate themselves through their approaches to QoS and pricing, in turn giving customers wider choice.

We argue strongly that the industry should not be aiming to have a single standard service plan — Diffserv is just a step towards more choice, not the end of the road. We believe that a proliferation of service plans will represent useful market differentiation, not just gratuitous attempts to achieve identical results using trivially different techniques. For instance, although the Integrated Services Architecture (Intserv [13]) has been shown to be unscalable across backbone networks [6], the service guarantee it offers would still represent a valuable proposition if it could be deployed scalably (it can using the M3I Architecture — see [35]). And the point is that a guaranteed service and a statistically differentiated service *complement* each other — there is no need to insist the whole world chooses one or the other.

Kelly's sample path shadow pricing (SPSP [36]) presents yet another radically different proposition, giving the customer's software full control of network QoS. Again, we do not believe there is any need to argue that this approach is better or worse than Intserv or Diffserv. It is a fundamentally different proposition that some providers will want to offer, and some customers will want to buy. Because it is so rudimentary, SPSP is of particular interest to us, as other service plans can be built from it.

We believe each approach has its place, with Intserv being primarily useful for direct delivery of QoS to human users and Diffserv being best for human users within corporate organisations, while the greatest potential of SPSP is where 'software is the customer'. By this we mean software that can decide its communications priorities within its own logic on very fast time scales.

The aim of the M3I Architecture is to ensure that an internetwork of approaches can all continue to manage their own and each other's supply and demand, even though each may be operating at a time scale anywhere on a spectrum from sub-milliseconds to years. Allowing freedom to innovate with new tariffs needs no justification. Why we need the freedom to choose QoS technology is not so obvious. One of our insights is that the choice of QoS signalling technology determines the balance of control between customer and provider, primarily *because* of its time scale assumptions. One provider may offer customers the flexibility to take fine-grained control themselves. Another may choose to take control on behalf of customers who don't want the bother of doing it themselves. Both are equally valid commercial approaches. Rather than embed such 'political' decisions in technology, we intend to allow choice of technology (as well as of tariff) under the same policy control.

So we have no reason to stop at three approaches. We have proposed yet more alternative service plans within the M3I project that each have their own particular attractions, and our intention is to enable others to continue innovating with further service plans. However, it is in no way possible to innovate without considerable practical constraints. We propose structural principles, which will ensure future flexibility is still possible. These principles in themselves constrain current designs (e.g. apparently ruling out roaming or carrier selection), but we give examples of ways to achieve the same ends without breaking the principles, and without having to resort to a regionalised architecture, as proposed in the original NewArch outline of requirements [14]. Instead, we show how different policies can synthesise different behaviours at the network edge, as long as certain principles are adhered to for core network design.

The Architecture is delivered in two parts: Part I "Principles" (the present part) and Part II "Construction" [20]. This first part is briefer and more discursive, justifying the architecture by focusing on the more difficult requirements and how it meets them. The second part is longer, being a full technical specification of the architecture. A brief summary of Part II is given in Part I, allowing readers who are content with a stratospheric

view to skip the second part altogether. In this spirit the conclusions for both parts are given at the end of this first part.

The technical side of the M3I architecture in Part II is designed to allow a network provider to choose any network quality of service (QoS) technology and set any tariff for it (that is to choose any *service plan*). However, if the service plan fails to take account of the fundamentals we outline here in Pt I, no amount of good architecture can save it. Further, if the design principles are not followed, it will not be possible to evolve to new service plans over time. Thus, the complete architecture is both a set of building blocks, and a set of principles to guide their construction. In fact, the Architecture encompasses customers' reaction to a service plan too. So any provider who fails to use the Architecture's capacity to evolve, will find customers exploit the Architecture to switch to more innovative providers.

The body of Part I below consists of just three main sections. First, the Fundamentals section outlines the set of principles that guide how to use the Architecture. Secondly, the Resolution of Tensions section is necessary because in practice it is never easy to apply absolute principles. So we describe how the principles have been interpreted in a few exemplary cases that were designed, built and studied during the course of the M3I project. The third section is where the brief summary of the detailed M3I Architecture in Part II can be found.

The M3I Architecture is the result of the considerable insight gained by the collaborating partners during the course of the M3I project. It is an evolution of the architecture used throughout the M3I project. It has been considerably improved and clarified due to the experience gained applying it to a number of example scenarios [1]. The contributions of particular individuals are acknowledged at the end.

Section 2 on Fundamentals follows below. It targets only the controversial aspects of a principled approach to market control under the four headings: Market Structure, Network Economics, Quality of Service and Connectionless Service.

2 Fundamentals

Rather than listing universal truths (e.g. the architecture should be secure, manageable, etc.) we focus exclusively on controversial and sometimes subtle principles. We expose the underlying linkage between matters of technical controversy and matters of commercial and even regulatory policy. These tend to be debated as issues of technical merit, when perhaps the real debate has an unconscious policy bias. The areas we cover include:

- Open market structure — where our technology allows closed, integrated markets, but only by choosing to apply a policy, which temporarily closes up an open design.
- Network economics — where we bring out the pros and cons of various fundamental ways to control fast variation in demand;
- Quality of service — where we focus on an emerging 'battle' between the networking and computing industries, the result of which will determine the evolvability of the Internet. We pinpoint how the time scale constraints of the different QoS signalling technologies reveal their 'political bias' in this battle, and give strong recommendations on how the network industry can either stifle its own evolution or *enable* interworking and choice, but without *having* to offer that choice until it is commercially advantageous. We summarise the results of our investigations into scenarios that become possible if the evolutionary path is chosen, in particular recommending how *not* to price QoS in such a world.
- Connectionless service — where we articulate principles that will enable a greater range of QoS control than imagined today, *and* allow full commercial flexibility, but by embracing connectionless (even flowless) working, rather than fighting it.

2.1 Open market structure

The first design principle we recommend is to conceptually **decompose** the market into its minimum component businesses, so that the components of the design can be re-composed in arbitrary combinations for full flexibility. This is in strong contrast to traditional telecommunications industry structures which often standardise around an agreed vertical business model (e.g. 2G/3G standards, with implicit assumptions about roaming agreements, tariff constraints, control of subscriber identity modules, etc.). Further, when re-composing the components, it should be assumed that each component business is not inherently trustworthy, so mechanisms to re-build trust must be inherent in the design. This recognises the industry's competitive rather than public service model, also recognising the need for future business model innovation.

A specific aspect of the decomposition principle argues that any technology, function or protocol that implements commercial decisions should be abstracted from the network infrastructure or protocols, or any other more generic part of the system. Rather such functions and protocols should be placed at a **policy** layer (conceptually between the user and the top of the application layer) so that it is easy for providers to differentiate themselves and customers to switch between them. This principle is primarily required because of the long time-scales necessary for infrastructure changes and their standardisation. Ideally, the whole charging system should be as separate as possible from the transmission system, while policy control of each should be separate again. Pricing should merely be *applied* to events already occurring in the network, rather than introducing elements into network protocols for charging purposes.

Secondly, we recommend use of the **end to end design principle** [50], to push specific capabilities outward to the edges of the network, preferably to customers themselves if trust allows, so that they they can innovate in the creation of new behaviours and hence new services, while the network remains simple and generic. This principle is applicable where a large capital investment is planned in a generic capability (e.g. the network), which is intended to outlive the requirements known at the time it is built. The skill in applying this principle is to push features outward that may seem generic, but in fact are not. For instance, the original design of the Internet recognised that the need for a connection was not a universal requirement. The solution was to delegate creation and control of connections to the end-systems that were using the network — a highly radical step at the time[1] (Section 2.4 discusses implications of the connectionless service model on the M3I architecture).

[1] In fact the end to end principle was articulated in hindsight about a decade after it had been applied in this case.

Recent research [30, 22] holds out the equally radical prospect that differential quality of network service (QoS) could also be controlled by end systems rather than the network. Such an approach would allow QoS requirements to continue to evolve, whereas investment in controls embedded in the network to meet the currently understood set of requirements, might unduly restrict future possibilities. For instance, most of today's QoS requirements centre around streaming media applications attended by humans, with little focus on, say, unattended processes with highly variable loads and priorities, that the application itself may not be able to predict.

The M3I architecture has been deliberately designed to encompass both end-system controlled and network controlled QoS, dealing with the many implications that would arise if both co-existed. We show how an internetwork designed on end to end principles of end system control can be deployed to support network controlled models, whereas the reverse is not possible.

A brief motto summarises these first two principles: "**minimise then synthesise**". Specifically, the end to end principle implies minimise in the middle then synthesise at the edge.

Thirdly, we recommend **split-edge pricing** [51, 17], which is analogous to encapsulation in software engineering. The price a provider offers its customer for service on one side of its network merely *takes account* of onward costs from onward networks in providing the service, but onward charges are not passed on unaltered directly to the customer. Instead, it is the provider's business to consolidate its pricing schedules otherwise they become increasingly complex the longer a network path continues.

The consequence is that each contract-related relationship (pricing, responsibility for service failure, charge advice, etc) must be **bipartisan**, that is between a single customer and a single provider, for each invocation of service. No provider should be able to blame a more remote provider for charges it is passing on, or for some technical requirement that another provider imposes. This is the reason this principle is so important to us architecturally — it minimises cross-industry standardisation linkages allowing providers to innovate in service offers and tariffing. If it is not applied rigorously, innovative tariffs and business models have to be taken through standards bodies — effectively stifling innovation.

In the following two subsections, we decompose the structure of the communications industry both vertically (between different layers of the communications market) and horizontally (between competing and co-operating providers within the network layer, which is our focus). The 'minimise then synthesise' principles are discussed in the context of vertical and horizontal decomposition, while discussion of edge pricing and bipartisan contracts is specific to the context of horizontal segmentation.

2.1.1 Market layering

An excellent description of the Internet industry structure is given in [3]. The M3I architecture focuses on the network layer of the communications market. Here we describe the essence of this layer and its supply chain interfaces both from below and onward to higher layers.

The network layer delivers service to the end customer's transport layer. The pricing and service from the layer below is considered *relatively* static within the time-scales that M3I technology adapts. Of course, it changes on longer time scales, but this has not been the focus of M3I. By longer time scales, we mean days to months, but in the future, the supply cost from lower layers might possibly change in seconds to minutes [38][2]. In fact, some lower layer bandwidth is already being auctioned to network layer providers on time-scales of the order of minutes (e.g. Band-X [7], Arbinet [5]). On the other hand, M3I technology *can* signal price changes from the network layer to the transport layer in packet time scales of milliseconds or less if required.

To put M3I in context, Fig 1 shows the main decomposed parts of the communications market, with arrows showing the direction of provision of service from one to the next. Particular aspects to note are that the physical and link layer markets only sell upwards to the network layer, which, by definition, is where links are connected together into a network. Physical links don't always map directly onto logical links in the link layer, the mapping being re-configurable in time scales of minutes to days. Some links that connect two networks are sold on a 'half circuit' basis to both ends. Others are sold to one end who re-sells use of the link to the other end (more common in the retail market, where some ISPs bundle both network and link service for their customers, e.g. the access link shown on the right). This is an example of re-composition of decomposed

[2] Kirkby uses shadow pricing for distributed resource control of a single provider's resources without explicitly implying the price could be used as a true market price between businesses. However, with the addition of accounting, such pricing could be used across a market interface.

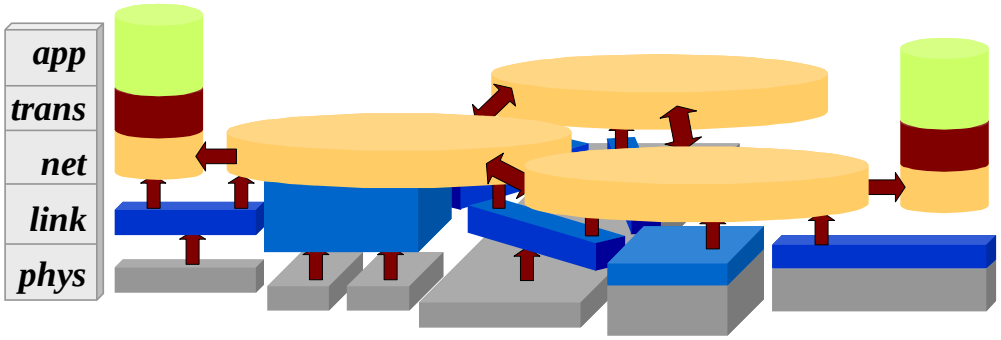

Figure 1: Vertical market layering and horizontal segmentation

minimal business components into a new business model.

It is no co-incidence that business boundaries tend to occur at the service access points between the layers of the Internet Architecture shown on the left. This is because, in a commodity market like telecommunications, it is near-impossible to place a business boundary where there is no clear technical interface. Thus the business boundaries of a minimal Internet network provider match the technical service access points above and below the network layer of the Internet Architecture (IP).

Edge network providers sell service to end customers (host communications stacks are shown at the extreme left and right of the figure). Alternatively, network providers sell service to virtual ISPs (not shown), who deal with retailing their service to end customers.

If applied properly the decomposition principle allows a business to choose to integrate together a number of functions in a closed way (e.g. to protect itself when it first enters the market). Then later it can open up its internal interfaces by choice, perhaps as the lowest level functions become commoditised. That is, for full flexibility, the system should be open by design, but may be closed or partially closed by policy choice.

The figure shows examples of such vertical integration, where two layers are not separated by arrows denoting sale of service across a commercial boundary. It is common for operators to integrate running a network with running the links of which it is composed. Similarly, the network and transport layers are often supplied by the same operating system vendor and the application may also be vertically integrated, as shown in both cases in the figure.

However, vertical integration is not confined to lower layers. A trend towards vertical integration of trust-related aspects of the application layer back into the network layer business is described by Clark & Blumenthal [23]. If this were shown in the figure, each network provider would support a transport and application layer — counter to the end to end principle, which confines these capabilities to end systems. They point out that the Internet has been the victim of its own commercial success. For some network users, so much is at stake commercially, that it is no longer possible to trust them to co-operate with the goals of running the network. For instance, it is generally held that customers cannot be trusted to only use QoS when it is really required. Because QoS is flow-based, this has led to transport layer capabilities such as flow processing and admission control operated by network providers. Further, application layer capabilities such as policy based admission control [55] are also being deployed in the network. The authors focus on vertical integration for other reasons like privacy, rather than QoS, but their conclusion applies equally if QoS is embedded in the network architecture: stagnation of the industry's innovative potential will be an inevitable consequence.

The work in the M3I project has shown that it is feasible to **synthesise QoS** at the end systems. We use pricing incentives to buck the trend described by Clark. Thus, at least in the case of QoS, we can preserve the flexibility of an end to end architecture even in a hostile environment of distrust. However, Clark implicitly argues that this is still not enough. Providers will still choose to implement network-based QoS in the short term, in order to add value to their service. In fact, each provider is planning to offer QoS in an isolationist way (or 'Balkanisation' as it is popularly called) as a competitive feature.

The M3I architecture deals with these tactical pressures more strategically. Balkanisation will bring short term profits, but in the long run, we predict that the value of globally interconnected QoS will win out (the value of

a network grows with the square of those connected — Metcalf's Law) [45]. Therefore M3I's focus has been on showing the feasibility of an end-system controlled QoS architecture, then we have worked back in time to synthesise shorter term business models. Thus we make vertical integration optional, rather than inescapable for deeply embedded technical reasons.

Moving from vertical segmentation to horizontal, note that the figure shows different providers across the network layer of any end to end path, selling their services to each other (double ended arrows). Principles on which to base this interconnect market are the subject of the following section.

2.1.2 Layer segmentation and edge pricing

Network providers both compete to serve the same geographical coverage as each other, and they co-operate to interconnect their different geographical coverages. Competition occurs as much in the interconnection market as around the retail edge of the network, with 'customers' comparing offers and switching between providers. Competition between providers can be simply on price using the same QoS/tariff service plan, or, particularly in the retail market, between distinct service plans that are less easy to compare directly. Co-operation to supply connectivity along an end to end path can involve interworking between different QoS/tariff service plans. The M3I architecture has been designed both for value comparison between competing offers and for gateways between co-operating service plans on a path.

In the introductory passage earlier, we asserted that split-edge pricing minimised inflexibility caused by the need to standardise commercial service plans. Using Fig 2, we will now justify this assertion. A unicast path across a network is modelled as a communications object shown as the all-encompassing circle with two interfaces. The service access points to the network layer on the hosts at each end of the path are analogous to the two outer interfaces shown, the one at the local end being e. Encapsulated within this model of the whole Internet path are a number of concatenated network domains (the medium sized circles). Encapsulated within each of these are a number of concatenated links and routers, and so on. Two alternative paths are shown, one starting with edge provider interface e_1 the other with e_2, who are competitors.

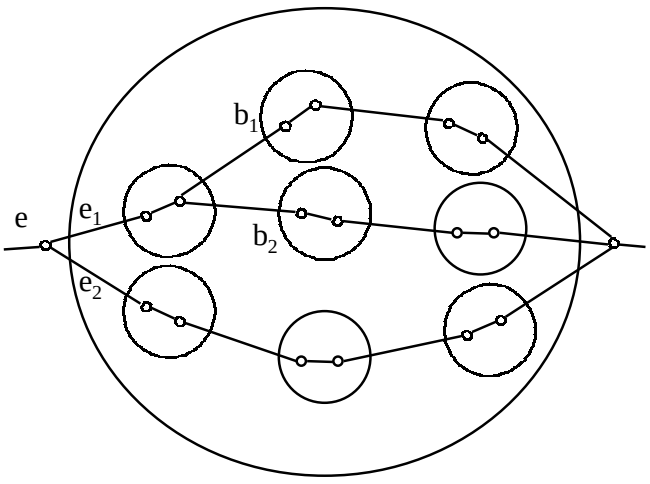

Figure 2: Encapsulation analogy of edge pricing

In a standardised industry like the Internet, one would expect e_1 and e_2 to be functionally identical. The Internet service interface, e, is implemented in the operating system of the user's device, common to all ISPs. But if we also consider incidental or 'non-functional' aspects of e_1 and e_2 such as contractual aspects, we would expect them to be potentially different. For instance, ISPs have different pricing plans. They also supply their customers with CD-ROMs which can contain the client software to access proprietary aspects of their service, such as initial contract set-up, charge advice, multimedia help-desk access, etc.[3] So commercial differentiation

[3]Of course, as it is a commoditised market, popular operating systems implement a basic standard contractual interface, so that basic ISPs can just supply the specific numbers to dial, addresses to access, etc. However, in

in the industry is possible by supplementing interface e.

Now let us consider what would happen if a provider broke the edge pricing principle. Specifically, if a provider offered service to a customer, which depended on the customer also having a contractual relationship with a third party. An example that contravenes the '**bipartisan contracts**' principle is user control over backbone routing, termed loose source routing, or '**carrier selection**' in telephony terms. Referring again to Fig 2, this involves the choice between internal backbone interfaces b_1 & b_2 being presented at interface e_1. The scalability of the Internet (or any) network architecture, relies on hierarchical routing, which carrier selection would break. Not only does e_1 have to implement a method for the user to select between the two backbones, but it has to give all the routing and pricing information to enable the decision to be made. So, even if the route were selected by pinning it to just one or two points [28] rather than allowing every possible inter-domain choice (five on the average Internet path with nine not uncommon [41]), knowledge of routing state at vast numbers of intermediate points would have to be exported to the network edge. Worse still, if the retail pricing deal were in terms of, say, traffic volume, the wholesale deal would have to be couched in the same terms to be understandable to the retail customer. Then, if wholesale pricing between the two providers were in terms of traffic volume pricing:

- either *all* retail deals depending on it would have to be in terms of volume pricing too,
- or every type of metering used at the retail interfaces would have to be applied in parallel at the wholesale interface, just so that each customer could make carrier selections in terms they could compare with their chosen tariff.

In fact, carrier selection is merely a work round for an inherently uncompetitive market. What is really required is good technology at the retail interface to dynamically choose between e_1 & e_2 — effectively better retail routing [26], not exposure of wholesale routing. Then if an edge provider is choosing an expensive backbone for its own commercial reasons, but against the interests of its customers, it will soon lose its customers to competing edge providers. Those who choose the cheapest backbone routes without being coerced by carrier selection will ensure carrier selection isn't required.

Of course, even if there are no technical barriers to a customer quickly switching to another access provider, each provider is free to put up commercial barriers to switching, such as a high contract set-up fee with low usage charges. We are not arguing against *commercial* barriers, because they can be chosen freely in a competitive marketplace. We are merely saying that the underlying *technology* shouldn't impose barriers to fast provider switching.

International **roaming** agreements in the mobile industry are another example that breaks the bipartisan contracts principle. They involve mutual agreement between edge providers that the metering aspects of interface e_1 will be able to support e_2's capabilities and vice versa. Metering is a commercial capability that constrains the possible set of tariffs. Therefore such a roaming agreement breaks the edge pricing principle. Assume one edge provider wants to introduce an innovative tariff, perhaps where the customer is charged based on reservations, or based on content downloads from certain sites, or giving refunds for downloaded adverts, etc. It has to choose between not offering the new tariff to roaming customers or submitting the required metering capability for standardisation across all its competitors, thus destroying any advantage of the innovation.

We are not saying that neither carrier selection nor roaming should be offered — our aim is to allow complete freedom in what is offered. We are simply saying that it is in everyone's interests to keep rigorously to the principle of bipartisan contracts, so it is worth taking care to set up business models that achieve the intention of carrier selection, roaming etc, but avoid stifling the industry in the process.

For instance, we can offer a number of alternatives to the current international roaming model which keeps to the bipartisan contract principle, but still gives customers the appearance of a seamless contract. In one example, a virtual provider presents a common tariff to its customer, but pays wholesale charges to access providers used by the customer against their own various tariffs. The virtual provider might always charge the end customer on the basis of data volume with time of day pricing. It would set each price per MB to cover the charges that typical access patterns lead to. It loses money on some customers and gains on others, but taken overall its price per MB is arranged to make a gain.

With split-edge pricing, any one network provider can act as the intermediary between multiple commercial 'standards'. The interface with one peer can comply with one standard, and that with another peer can comply with another standard. By 'standard' we mean a common type of service plan, such as charging based on

general there is a potential for ISPs to differentiate themselves commercially at this edge interface.

traffic volume, or on congestion, or on reservations etc. Internally, the network provider can intermediate all the different standards, because it can compare or add all the benefits and costs at each interface by folding them into the single dimension of monetary value [18].

Standards are still necessary in the industry, mainly for the benefit of equipment and system vendors selling across the industry. But split-edge pricing ensures the *commercial* interfaces of the industry can be based on multiple, evolving standards, not just a single increasingly archaic one. Also, when a standard needs changing, a 'flag day' isn't needed to change the whole world at once. The M3I Standards Activity statement [19] describes the minimal standards required to introduce the M3I architecture into the industry.

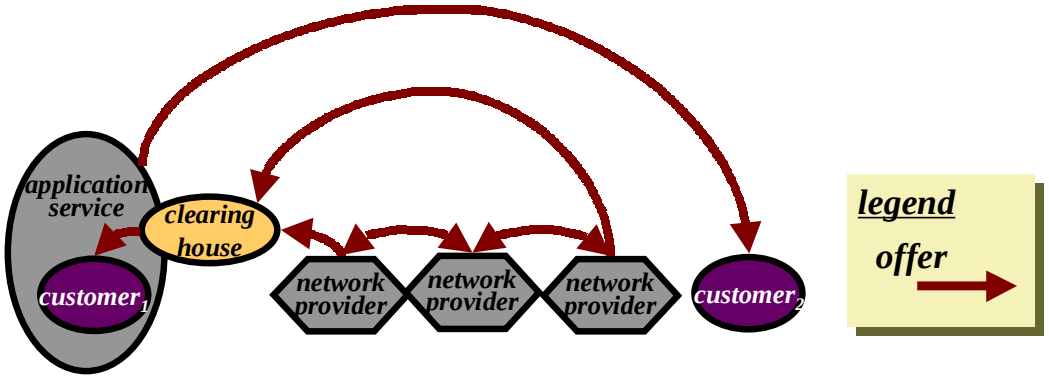

Figure 3: Clearinghouse

Reapportionment of charges between the ends of a communication is a third example that has potentially disastrous consequences for future industry flexibility, if designed without regard to principles. Examples from the telephony industry that would need this capability in the Internet are 800 (free-phone) or termination charges for calls from fixed to mobile networks.

If an access provider arranges to clear the charges for an end-user's session through the next hop provider towards the core of the Internet, it can be arranged for the money to eventually be distributed correctly across all the providers on the path, whether it originated from one end, or the other, or both. However, such an arrangement would require backbone providers to be able to sort packets crossing their network into buckets; one for each of possibly many reapportionment arrangements at the retail edge of the network. This would be necessary to know what share of the revenue for each packet is coming from which direction. Thus, the contract at a wholesale interface depends on all the possible arrangements at the retail interfaces (breaking the bipartisan contracts principle).

Fig 3 shows an alternative that only uses bipartisan contracts, thus avoiding stagnation into a few globally standardised reapportionment schemes. In fact, this model also offers far more flexibility to reapportion charges in novel ways than the inferior, network-provider based models. It involves an end to end clearinghouse role, which buys service from both the sending and receiving end, and re-sells the complete flow to one end or the other (or both in arbitrary proportions). This model is also extensible to flows with multiple ends (e.g. multicast). Reapportionment of charges is discussed at length elsewhere [18, 17, 3].

Of course, not all of a network providers costs emanate from its peers. It has operational and sunk costs from lower layers (the arrows in Fig 1) as well as its own internal costs. Again, the edge pricing principle states that it is not necessary to allocate these costs accurately depending on how much each customer uses (e.g. based on internal routing decisions). True-cost pricing is itself costly, therefore a provider is free to present prices at the edge of its network that make some convenient averaging assumption about cost allocation. Of course, this does make the provider vulnerable to arbitragers who can cream off low cost business, but the provider is always free to refine its cost allocation strategy.

2.2 Network economics

Using price to resolve the tension between supply and demand has been used by network businesses throughout the history of communications. However, demand has always fluctuated far too quickly for the supply of new capacity to keep track. Regular demand fluctuations (e.g. daily ones) have been damped with regular price variations (e.g. time of day pricing) in order to better match supply to demand. But, not only does each customer's demand rise and fall rapidly, it also moves instantaneously from one network path to another, making demand for any one network resource highly volatile. So, demand is still too variable to guarantee that the prevailing price will be able to cap it below the deployed capacity. Therefore, admission control has always been necessary to block the worst peaks in demand — that is admission control at each resource everywhere in the network. Generally the alternative of over-pricing or over-providing capacity makes a provider uncompetitive against others that are prepared to use admission control.

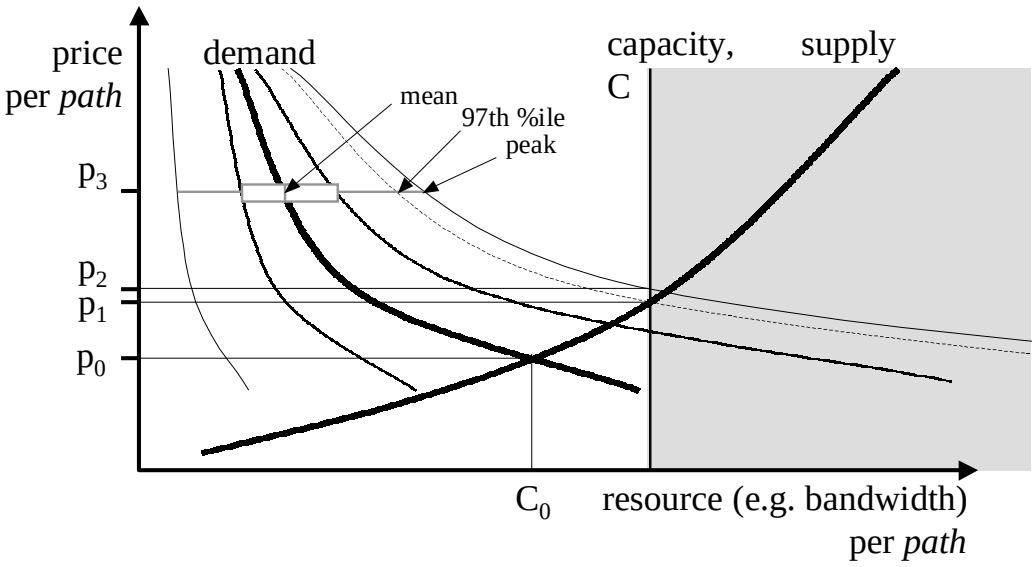

Figure 4: QoS economics

Drawing from classical microeconomics, Fig 4 illustrates this point, showing supply and demand curves against price for the resources on one network path. But it also shows how, at any one price, instantaneous demand varies widely. Thus, instead of a single curve, demand is better shown by a probability distribution across a wide range, illustrated by the 'box-and-whisker' plot at price p_3. For clarity, curves are plotted linking points of equally probable demand (e.g. mean, upper quartile, peak, etc.). To decide how much capacity to deploy, our example provider calculates its supply against price curve shown, based on the cost of capacity, its economy of scale and on likely competition. Even if the provider could perfectly predict the demand probability distribution from market analysis, it would not deploy the capacity C_0, where the mean demand curve crossed the supply curve. If the price charged was the one derived from this intersection, p_0, this decision would lead to half the demand having to be turned away, because of demand variability. Instead, this example provider has chosen to try to meet about 97% of demand. To implement this policy, at each point in its network, the provider has deployed capacity, C, by predicting where its long term supply curve will cross the 97th percentile of demand (shown dashed). If it then charges price p_1 it has to turn away about 3% of demand. This policy is equivalent to aiming for a 3% packet loss-rate for a connectionless service (or a 3% blocking ratio for a connection-oriented service). Incidentally, we are not implying that all providers should decide their position in the market this way — it is just an example.

In the short term, whenever demand is greater than the chosen capacity, only a maximum of C can ever be served. Therefore, if it could, the provider would like to set the price at p_2 during these episodes. It would still only serve capacity C, but at the highest possible price, thus maximising return from the available capacity. In effect, the price becomes the admission control mechanism, with the 3% most price-sensitive customers suppressing their own admission.

If we have a mechanism to vary the price as fast as demand varies, there is no longer a need to consider demand as probabilistically distributed (after having made a long term capacity deployment decision). Instead the instantaneous demand can be used to determine the price at any one time. That is, not only can the provider raise the price from p_1 to p_2 when demand peaks, it can also drop the price below p_2 when demand drops below the 97th percentile. This would make commercial sense in order to maintain the demand share in a competitive market, particularly if demand could be increased proportionately more than the price was dropped — high demand elasticity.

In fact, if there *were* a dynamic pricing capability, the scenario could be radically different from the outset [40]. Capacity would not need to be deployed to cater for near-peak demand in the first place. Instead it would be possible to just deploy C_0 for instance, and the price would fluctuate around p_0 rather than p_1 — cheaper than competitors on average. If price were used in this way, one could keep packet loss rates (or session[4] blocking ratios) down to zero. Under traditional definitions, this would also imply a drastic improvement in these dimensions of quality of service. . .

. . . However, the market analysis that determined these demand curves will have assumed that the price was effectively stable. In practice, varying the price dynamically to adjust demand, will itself depress demand. In other words people are willing to pay extra for a predictable price [46]. But the situation is no better with admission control and a stable price; demand is also depressed by unpredictable service availability [12]. So there really is no ideal solution: either the network drops 3% of packets; or the network rejects 3% of session requests; or the network price is too high 3% of the time, effectively making it unavailable. Ultimately, all solutions are as unpredictable as the onset of congestion, which is as unpredictable as the customers who cause it.

The big factor in favour of the dynamic pricing solution is that customers or their applications maintain full control. The extra decision-making required can be dealt with by an agent [27, 22], but the customer or the application can still override the agent's decisions, which is not possible with network-based admission control.

This shift of control is also important for the evolution of new applications. With network-based admission control, the user's application must have foreknowledge of its network requirements in order to form a service request (whether per session or per month). With control on the end-system, requirements can change arbitrarily frequently without extra messaging costs. Thus, admission control decisions can even be made within the complex logic of future applications like games, rather than assuming QoS requirements will always be restricted to static requests primarily suited to streaming media applications. In economic terms, one can measure the value that customers place on having control in emergencies [11], but it is hard to put a figure on the strategic value of deploying technology that enables unknown future applications.

To summarise, traditionally the packet loss rate (or the session blocking ratio) has been a large part of the quality of service of a network. If we include scenarios with price variation, we can fool ourselves into thinking we have dramatically improved quality of service [40]. Indeed, we have improved QoS, and we have given customers complete control of the trade-off between QoS and cost, but the unpredictable price we have used to achieve this detracts from our success. The question is whether the improved QoS and greater control add more value than the variable price removes. The only assessment common to these alternative approaches is their relative value to customers. Therefore it is difficult to devise an objective experiment to compare their relative merits. Targeted experiments on users have been conducted within the M3I project [31], but ultimately the only valid comparison between them is to see which one wins in the market-place, or at least to allow both to serve their particular niches.

Therefore the M3I architecture that follows caters for different scenarios where different *edge* providers choose to offer anything along the spectrum from pure congestion pricing to pure admission control. The architecture is therefore capable of instantiating one network technology and various tariff offers for one provider, and a different network technology and different offers for another provider. The M3I architecture therefore offers interworking between provider service plans, and practical guidelines on how to limit interworking complexity.

A market controlled architecture must also necessarily feed events to a charging system to police the market. We discuss the cost of charging and how to minimise it later, in Section 2.4.1. But first we go into quality of service in more depth, given its centrality to our theme.

[4] We use the wider term 'session' from Internet terminology, in preference to 'call' from telephony.

2.3 Quality of service (QoS)

Internet QoS is a broad subject, involving effects over a spectrum of time scales. Here we focus solely on general aspects of QoS that impinge on the engineering of a pricing system architecture. From this perspective QoS must be separated into two aspects:

- the best long term QoS an unloaded network can attain and
- short term QoS degradation during congestion.

The M3I architecture allows charging for either or both, but the most difficult issues to tackle are those to do with the fine time scale control system engineering, which has therefore been our main preoccupation.

2.3.1 QoS pricing

The M3I architecture is designed to be able to charge on any basis for uncongested service — volume usage, offered QoS, flat subscription or whatever — a price can be *applied* to any measurable aspect of the service. We strongly deprecate any moves to *add* capabilities to the network service purely for charging, for two reasons:

- embedding current commercial ideas into network infrastructure is likely to limit future commercial innovation potential
- if a feature is only necessary for charging, and not essential to the running of the service, it is very difficult to stop attackers removing the feature, thus avoiding charges but not affecting their service.

As flexibility is our goal, we try to avoid any recommendations about anything to do with the commercial aspects of a pricing plan. But in this section we make strong recommendations about the general form that QoS pricing should take. In particular, when pricing QoS degraded by congestion.

A common misunderstanding stems from the apparently natural idea that the better the QoS the higher the price should be. This is a reasonable position when pricing *offered* QoS, but it is fundamentally unsafe to network stability if setting the price of *delivered* QoS. Offered QoS is synonymous with the relatively static, uncongested QoS we introduced in the previous subsection. Delivered QoS is the result of degrading static offered QoS with congestion.

If the price goes down with congestion, there is a grave danger of congestion collapse. If pricing is to be used to encourage self-admission control, the price should rise as congestion approaches, preferably *before* it starts to have an effect on QoS. The idea is to push back demand to *avoid* congestion, but without having to charge more for poorer service. If congestion continues despite continuing price rises, QoS will eventually suffer. But the price must not be dropped (nor refunds increased) faster than the value of the service drops, because demand will then increase, leading to collapse.

Of course, if session admission control rather than price is used to protect against congestion, the QoS of successful sessions will not be degraded by congestion, and the price can stay constant as demand varies.

2.3.2 The Essence of Connectionless QoS

Below we describe the essence of QoS in order to support the strong statements above about correct and incorrect ways to attach pricing to QoS. Our description of the essence of QoS also provides the vocabulary for the brief but fundamental comparison of QoS technologies that follows.

Fig 5 clarifies some common misunderstandings about the quality of a connectionless network service. It shows the characteristics of the *physical* network resources in the centre, with the characteristics of the *traffic* applied to it on the left, and of the *traffic* delivered from it on the right.

The first insight is to qualify bit rate as three different things, only one of which (the first) is a measure of QoS:

- available network bit rate along any one path — the relatively static quality of the network service;
- sent traffic bit rate — a measure of the load applied to the service, not of the service itself or of its quality (similarly for the source's burstiness);
- received traffic bit rate — not a *quality* of the service, but the delivered service itself.

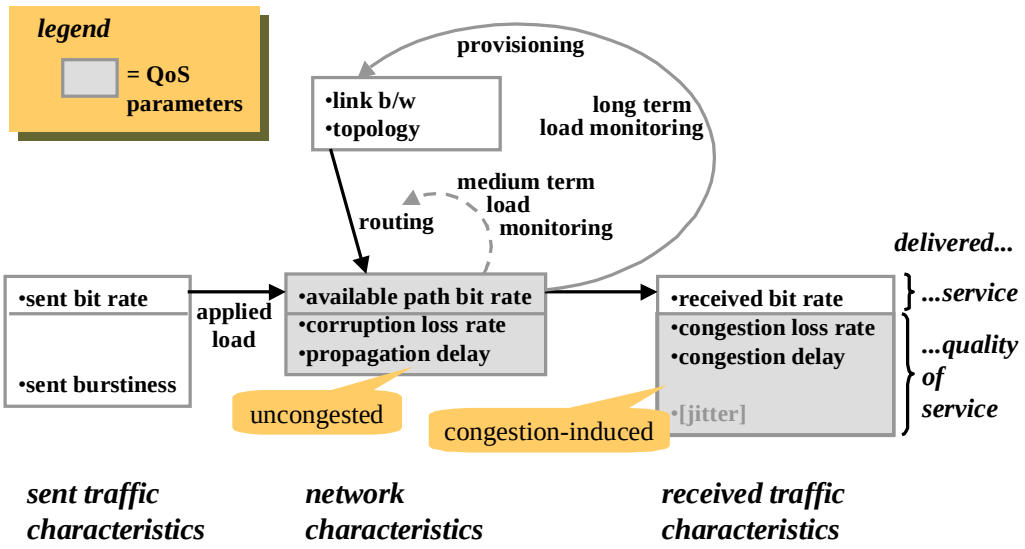

Figure 5: Connectionless QoS

Uncongested QoS Along with available bit rate, the other main relatively static quality of the uncongested network service is the propagation delay along any one path. The inherent loss rate on the path is a third static quality, but only becomes significant whenever wireless links, with their inherent background interference, form part of the path. These three parameters, which are typically the result of longer term routing and sizing decisions, completely specify the QoS of the network and any path through it, *in the absence of congestion*.

Congestion-induced QoS If total bit rate sent from many sources through any one network resource is greater than the available bit rate, congestion is defined to have occurred. This degrades other qualities of the delivered traffic flow:

- congestion-induced delay — caused by buffering during short episodes of congestion[5] and by the need to retransmit essential data lost during longer congestion episodes;
- congestion loss rate — caused by buffer overflow, spectrum congestion etc.;

2.3.3 Control assumptions in QoS technologies

In this section we expose the implicit assumptions about industry structure, control and economics that are embedded in the main QoS technologies and even more so in the 'architectures' that have been built around them. Once these implicit assumptions are clear, each approach can then be measured up against the fundamental principles given earlier. Critiques of QoS architectures in terms of their political outlook are rare — they are usually compared on technical practicality grounds. Our aim is to help decision makers (policy setting organisations as well as providers) to understand the constraints that design choices will have on business policies and future industry flexibility.

The QoS signalling technologies we consider are the resource reservation protocol (RSVP [15]), the differentiated services code point (DSCP [44]) and explicit congestion notification (ECN [49]). The architectures we consider are the Integrated Services Architecture (Intserv [13]) and the Differentiated Services Architecture (Diffserv[6] [24, 9]). ECN is not yet accepted as an integral part of any coherent QoS architecture — an omission we aim to remedy here.

[5] Delay variation causes jitter, but maximum delay is all that is important as jitter can always be removed by receiver buffering, therefore jitter is not treated as a distinct QoS parameter.

[6] Note the general term differentiated services (lower case) just means services that are differentiated, whereas the term Diffserv is used for the specific Differentiated Services Architecture (upper case) used for differentiating services.

One of the results of the M3I project has been to show that there is no *need* to standardise around QoS 'architectures' like Intserv and Diffserv. Standardisation is useful for equipment vendors, but ultimately anti-competitive if all operators apply the same business model. These 'architectures' are merely useful as exemplary business models to demonstrate that a particular QoS signalling technology can be applied commercially.

The underlying QoS signalling technologies, not these 'architectures', are the important 'molecules' to standardise. We believe the addition of ECN creates a sufficient toolkit of signalling protocols that should be universally supported. The tools in this kit can be combined (and have been in M3I) to create new QoS 'architectures' that can reflect the balance of control that each provider wishes to promote. The provider describes which control architecture it is promoting in its tariff, by applying prices to the QoS signals it deems significant. Therefore, it is more important to **standardise tariff** description languages and protocols, at least to avoid *too* many alternatives. As already explained, we term this combination of tariff and QoS technologies a 'service plan'. The M3I project has investigated a number of scenarios, by inventing novel but realistic service plans, many examples of which are given in Altmann *et al* [1]. We introduce some later in this document to illustrate various points.

From the insights on the essence of QoS above, it can be seen that the network provider is inherently best-placed to control resource allocation (and hence uncongested QoS), while the set of data senders have inherent control over offered load (and hence congestion-induced QoS degradation seen by receivers). Each QoS signalling technology is suited to signalling at different time granularity. Between signals, the party with inherent control is in charge. QoS signalling allows a party *without* inherent control to influence the party *with* inherent control.[7]

Consider time granularity on a spectrum from per contract (weeks or months) through per session to per packet. We can place each QoS architecture's assumptions about who is in control on this spectrum. Although signalling allows control of the system to be shared, the longer there is between signals, the more inherent control becomes actual control in practice.

Requests for Diffserv traffic conditioning agreements (TCAs) typically cover periods of weeks. Intserv reservations are typically per session. While packets themselves are each effectively service requests at sub-millisecond time scale. These are all examples of request signals. But in fact, three types of signal are necessary: requests, responses and advisory feedback. Each may be explicit or implicit. Responses may be acceptances or refusals. For instance, sending a packet is an implicit request; dropping a packet is an implicit response (made explicit for the sender by the receiver). Incidentally, these two are the only signalling mechanisms of traditional best effort Internet service, which operate purely at per-packet time granularity.

Intserv introduces both requests and responses on per session time scales. During a session, the provider assumes control of quality, at least for the reserved traffic. Any congestion will be due to unreserved traffic. Thus it is completely safe for the provider to assume responsibility for control between signals because there is no possibility of packet-level refusals within a reservation. However, this makes Intserv unsuitable for applications where the path or approximate traffic level cannot easily be predicted at session start, limiting it to streaming and the like.

Diffserv introduces a conflict between acceptance and refusal time scales. Certainly requests can be denied at the time they are made (per contract duration). But, even if a contract request is accepted[8], packet-level requests may be refused within the contract period, during times of unforeseen congestion. Specifically, the level of service offered to packets of one DSCP may be congested, causing even traffic within contract to have to be re-marked to a lower DSCP, or dropped in the extreme (annotated with '!' in Fig 6). Thus, Diffserv is at the extremely customer-hostile end of the spectrum, more appropriate for very conservative providers[9]. Diffserv would be chosen by providers who are only willing to take control of provisioning for uncongested QoS if their customers assume all the burden of traffic planning, without even offering any guarantees on congestion-induced QoS in return.[10] An analogy would be an airline that required potential customers to pay in advance for the maximum number of flights they expected to require throughout the year, but wouldn't give any guarantee of a place on any of them.

We now note that neither Diffserv nor Intserv give any advisory feedback to *warn* of current network status, as

[7] Pricing is an easy way to add substance to this influence, but we are trying to expose implicit control assumptions that are independent of pricing models.

[8] Assuming it has a statistical element, e.g. an 'all addresses' TCA.

[9] A sad recognition of what was necessary to get QoS deployed?

[10] Of course, competition injects some discipline to ensure providers try to size so that congestion rarely occurs. Automatic refunds during congestion have also been proposed, but they introduce enormous complexity and can easily become incentive-incompatible.

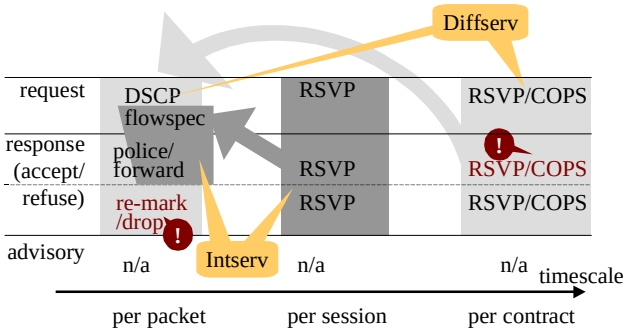

Figure 6: QoS time scales

distinct from simple binary responses to service requests (the bottom row of the figure). For example, ECN is a packet time scale signal warning of congestion levels on the path in use. So neither Intserv nor Diffserv can exploit any capability of end systems to adapt to congestion. This is a reflection of the control assumptions in these two architectures. Both architectures focus on the provider's inherent control over uncongested QoS. Neither makes any attempt to exploit the inherent control of an elastic sender over congestion-induced QoS[11]. Thus, neither architecture can claim to be designed under end to end principles. Choosing either architecture is therefore more likely to have negative consequences for future innovative potential. In contrast, providers who offer congestion notification from their network are inherently implying that they believe end systems have a role in controlling QoS.

Intserv prevents congestion by admission control, so ignoring congestion control is at least consistent, if a waste of a useful degree of freedom. However Diffserv just tries to ensure congestion will be rare, relying on good sizing by aggregating knowledge of all traffic conditioning agreements. Therefore, it would be sensible to exploit congestion notification technologies alongside use of Diffserv. We must emphasise that this is an omission of Diffserv (the architecture), not of the DSCP signalling mechanism as a technology. For instance two scenarios considered within the M3I project allow the DSCP to support end to end control — the User Direct scenario [1, 2] and the combination of ECN and DSCP behaviour in [21].

The industry's focus on provider-controlled QoS is also a symptom of a preoccupation with fixed networks, not **wireless**, despite its likely future predominance. Even if a provider has secured inherent control of the resources used by a wireless link (e.g. by buying a spectrum license), it still doesn't have control of the vagaries of the radio *environment*. So, if there is at least one wireless link on the path, it is important for the sender to be able to adapt to prevailing conditions, even if the provider is doing its best to control quality. The adaption required is similar to that needed to react to congestion.[12] If the end systems will have to adapt anyway to the resulting quality delivered from an end to end path, QoS mechanisms that co-operate with adaptive applications will be more fruitful than ones that attempt to offer guarantees assuming applications will not adapt. This argument is strongly reminiscent of the original arguments for end to end retransmission in TCP/IP, which was being done anyway so making hop by hop loss detection redundant.

To summarise so far, the omission of congestion control mechanisms from the predominant QoS architectures might reflect prevailing industry assumptions about who should control QoS. Alternatively, it might reflect a provider's deliberate policy choice to over-provide capacity, avoiding the need to consider congestion as a business issue. Whatever, such a decision is a policy choice, which may become outdated by events (e.g. if over-capacity becomes uneconomic relative to the cost of usage-charging mechanisms, perhaps because the cost of wireless spectrum dominates). In the longer term, QoS could become a commodity, and some providers may *choose* to realise its greater value by giving its control to end systems, allowing them to innovate with it (end to end principle). The M3I architecture recognises congestion signalling as an essential *technical* element in the QoS toolkit, but still allows each provider to express their *policy* on where control should lie. Such a policy is naturally expressed in a provider's offer of service, in particular in the tariff. Each provider will apply pricing

[11] Diffserv doesn't say congestion control should *not* be used, it just treats it as an orthogonal capability to QoS — for instance it is perfectly valid to use Diffserv and TCP congestion control.

[12] The incentive structure is very different though - a sender can cheat against other senders, but not against the environment.

to the signalling protocols that it has arranged to affect its service. The M3I architecture therefore ensures **no single, standard industry control model** is required, allowing evolution through market competition, going beyond a mono-culture of Diffserv, for instance.

QoS and price Stability We can now illustrate how the above control time scales constrain business flexibility, focusing particularly on stability of quality and of pricing. Our user experiments [31] and the literature [10] have shown stability of price and QoS in particular, are highly important to network customers. On the other hand, flexibility to vary either would be highly desirable to a network operator, in order to manage load.

QoS can only be guaranteed to be stable across the time span from one accepted request to the next possibility of a request denial. With Intserv, this means QoS will always be stable during a session. But QoS across multiple sessions cannot be guaranteed, as any one session may be refused. With Diffserv, QoS should usually be stable across multiple sessions, but on rare congestion occasions QoS may degrade unpredictably.

Pricing can be arranged to be stable over any duration. On the other hand, pricing cannot change more frequently than the timescale of any of the three types of QoS signalling (as there would be nothing measurable to attach the price to). With Intserv, this means that the pricing of the reservation cannot vary within a session. With Diffserv, pricing cannot easily be varied within the duration of a contract (except perhaps through refunds triggered by failures to meet the contract).

Therefore the choice of QoS signalling technology fundamentally constrains later choices over QoS and price stability. We will come back to this point later when we show how a provider can emphasise the use of different signalling technologies, thus moving its business model around the axes of QoS stability and price stability, purely using the policy control of tariffs (Fig 9).

2.4 Connectionless Service

The network layer service of the Internet is fundamentally connectionless and strictly it is meant to be flowless. But packets invariably include flow/connection identifiers, primarily intended to be used by the end-systems. The network provider cannot guarantee to be able to rely on these identifiers, because they can legitimately be spoofed or concealed by end to end encryption (e.g. IPsec) or simply making some out of band arrangement between the ends to change the meaning of a flow identifier. This fact fundamentally constrains the two aspects of network service central to our subject: network QoS and network charging. This section clarifies the nature of this constraint and summarises its architectural implications.

The main point to note is that Internet providers are free to offer service and pricing at the granularity of flows, but only if this is in the customer's interest. In order to exploit such a service, a customer must be willing to truthfully reveal flow identifiers, even if they would otherwise be hidden. For example, IPsec encrypts flow identifiers along with everything else within the IP payload, but they can be repeated in the clear in the IPsec security parameter index [8]. Earlier, in Section 2.3.1, we deprecated adding inessential information to packets as it poses a security risk. The problem is that revealed flow identifiers don't have to match concealed ones, which are the ones actually used for transport layer demultiplexing. So a network provider must be careful to ensure any flow-related service or charging cannot be exploited by concealed flow identifiers. The safest rule of thumb is to always ensure default service and pricing is for the bulk service to a network address irrespective of division into flows.

Below we discuss the apparent need for per-flow processing in the network in three cases: QoS, admission control and charge itemisation. There is a strong temptation for network operators to offer these services with per-flow granularity, but we caution them to proceed with care. Not only are there the above potential security flaws, but the added complexity in the network will limit future flexibility, as advised by the end to end design principle. So, in each case we point to alternative proposals that synthesise the same capability on the end system. Another way of looking at these alternative proposals, is that they can generally by-pass the network's ability to offer these services anyway, killing the case for investment in the equivalent network-centric capability.

QoS: QoS has only some relevance at a packet level (latency, instantaneous loss probability). Most QoS features concern statistical properties of flows of packets (available bit rate, maximum delay, mean loss rate etc.). If QoS is to be controlled in the network, the packet stream has to be classified into flows, which

gets more complicated the more possible ways there are to specify a flow (e.g. in AH, ESP, UDP, TCP etc.). Despite being network-centric, recent QoS technology proposals like Diffserv, aim to avoid per flow processing in the network, unlike Intserv. Of course, proposals for fully end-system controlled QoS (see for example the Dynamic Price Handler in Section 3.2) can use per-flow processing, as they work above the network layer, where flow-based control is meant to be implemented in the Internet Architecture.

Admission control: Admission control is primarily meaningful in the context of flows, not single packets. Many real-time streaming applications are 'inelastic', meaning that people would prefer to have x% of sessions blocked, rather than all sessions allowed but x% of packets within each session dropped. Recent research suggests that session admission control can be synthesised at the network edge by probing a path for its packet loss rate at the start of a session [29, 16, 37]. If successful, this line of research will remove another need for per-flow processing in the network.

Charge itemisation: Finally, charge itemisation creates a further need for per-flow processing in the network. For instance, a network provider might arrange for its local customers' charges for certain flows to be paid for by customers at the remote end of the network, on condition that signed proof was presented that the remote party had agreed to pay for each such flow. For instance, the remote party might agree to pay for all flows sent from the local network to any port numbers under 1024. If some flows between the two parties were encrypted, a malicious sender might label flows to port numbers above 1024, as if they were to ports below 1024.

Some hosts share one network level address across many users (e.g. a video hosting platform). If any of the users on such a host were using end to end encryption it would be impossible for a network provider to itemise the bill across each user of the platform. Instead, all the network provider can do is charge the platform provider the aggregate bill and any itemisation would have to be performed on the platform, based on metering in the operating system, above the encryption facility.

Therefore, general advice is for edge network providers to only charge to the granularity of network addresses, delegating any finer charge itemisation to the operating system of the hosts involved [22], unless there is a clear incentive for the customer not to cheat. At an inter-provider interface, we would advise avoiding per network address itemisation altogether for obvious performance reasons. Instead, bulk itemisation (e.g. by DSCP) is advised, but a certain amount of classification into groups of IP addresses may be necessary and feasible. The earlier section on edge pricing explained how to avoid the need to itemise flows at inter-provider boundaries (see e.g. Fig 3).

This leads us neatly into a discussion of the cost of mechanisms required to support market control, as promised earlier. It will be seen that this issue can also largely be resolved by keeping strictly to connectionless network principles.

2.4.1 The cost of market control

There is often concern that using a fine-grained market to control demand leads to excessive costs in order to manage and maintain the charging system. This fear is based on partial truth, but has been fuelled by an unfortunate legend that has built up concerning how much traditional telephony billing systems cost, which are often quoted without sources at between 30-50% of total telephony costs. This is of great concern, as one would expect telephony charging costs to be a lower bound on Internet charging systems, given telephony is a mature and less complex, but similar industry.

A good deal of cost modelling has been done in the M3I project, but mainly on the additional cost of introducing QoS management technology in the network, rather than on charging systems [39]. Our more informal investigations have found it is very difficult to get meaningful figures on the cost of charging systems, which are shrouded in secrecy. However, we have been able to find informally that these 30-50% figures seem to derive from considering *operational* costs only. However, when fixed costs are depreciated into the year's accounts, they dwarf these operational costs. The result when considering both fixed and variable costs is that charging and billing (both development and operations) seem to be about 4-6% of total costs, depending heavily on the policy on depreciation rate. However, there are huge caveats on these figures, as they are derived from very sparse information (no public sources can be cited, other than some tentative agreement in [42, 47] — 5% and 4-6% respectively).

If this new 'rumour' on costs is accepted, the order of magnitude difference from the previous 'rumour' certainly alters the approach one is willing to consider for controlling demand. Assuming that we have decided to use some form of charging for demand control, we still propose a principled approach is necessary, to contain costs as much as to prevent architectural violations, with the attendant risk of limiting future flexibility.

As we explained above, Internet service is strictly flowless, so we can expect complexity costs to rise if we continue to fight against the architecture and charge for such a service on a per flow basis. Further, incentives to **optimise utilisation** are improved if the granularity of the service matches that at which it is charged. However, *human* customers weigh the value of the service in units they understand, which typically means by the session. Therefore, we suggest a general principle should be that sufficient incentive to control demand can be given by **charging in bulk for packet service**[13] (as advised in the previous section but for different reasons). But if the customer needs her charges itemised into sessions, this can be offered as an *optional* extra, but only by the edge network provider. Because the customer has the inherent control to prevent *any* provider itemising some packets into flows (by using end to end IP payload encryption), per flow itemisation can *never* be the default anyway.

Looked at another way, itemisation should never need to be an essential part of the network as customers can always do it themselves with suitable software [22]. After all, end systems dealing with flows and the network dealing with packets sits comfortably with the Internet Architecture. And customers will always understand their own session granularity better than the provider. Of course, this principle also ensures compute and data intensive operations are distributed as far to the edge of the network as possible, which complies with more general **scalability** principles too.

However, there are implications for support costs if customers *have* to run itemisation software themselves in order to be able to receive charging feedback. On the other hand, a large part of the cost of managing a market mechanism is human intervention. In particular, collecting bad debts, managing customer complaints over billed items under dispute, etc. Itemisation can hugely increase such human support-related costs, but it also increases system costs by orders of magnitude.

This principle has been applied in the M3I project, where the provider's charging system has been built to be able to charge at a granularity dictated by policy [53, 54], not as the default. In scenarios where we have tariffs like congestion pricing, a 'mini charging system' on the end-system can work out sufficiently accurate charges to be able to react properly. The tariff gives the customer the incentive to run such software, which would anyway be impractical for the provider to offer, as it is part of the fast loop of congestion control. However, in most scenarios, the charging and accounting system is offered as a valuable, but ultimately optional, service.

[13] As long as we can assume the human customer has a computer to manage packet-level decisions, which of course we can.

3 Resolution of Tensions

In the original M3I requirements analysis [3] we characterised the problem we faced as a number of mutually conflicting requirements in tension. Our general approach is to allow customer choice at all time scales. But excessive choice and control is not always useful, because there are fundamental tensions between excessive choice and *human* limitations. Nonetheless, we believe that the problems of extreme levels of choice only apply to human-controlled traffic, with extreme choice becoming the primary requirement in the future, as traffic from unattended operation dominates the mix. So our overall intent is to solve the short-term problems that *humans* have in ways which still allow the evolution of longer term capabilities that *computers* will be able to take advantage of.

So we describe how the network architecture needed for this future can be deployed now, in such a way that customer choice and control can be limited at first, but progressively introduced. We describe the practical implications this has on choices of QoS technologies being made now. Then we describe how to resolve the human-related tensions to make this future network architecture useful for today's problems.

3.1 The evolvability of choice

A common theme throughout the preceding sections has been that the M3I Architecture allows each provider to choose its own set of service plans, so that the market will ensure the best service plans win, or that different service plans will be able to co-exist, each serving their own niche market. Thus, our glib solution to the problem of mutually conflicting requirements is an architecture that allows customer choice. But freedom cannot be given without responsibility. Choice cannot be between different control architectures in isolation, but must be intrinsically coupled with their pricing[14]. Of course, in a competitive market, pricing will reflect the intrinsic cost of each architecture, including the cost of charging for it.

But we discussed, in Section 2.2 on network economics, how choice can be much finer grained than just between competing service plans. The ultimate in fine-grained choice is for any sender to be able to send whatever it wants, at any time, on a packet by packet basis. To make the discussion concrete, we will now discuss how such an extreme in customer choice might be realised.

First, such a libertarian network would be completely out of control without an equally fine-grained responsibility mechanism. Kelly's sample path shadow pricing (SPSP) proposal [36, 30], creates a practical responsibility mechanism of the requisite granularity. The output queue process of every router randomly marks a designated field in each packet with greater probability the greater the congestion of the queue. Network providers apply a price per mark, which gives the receiver the incentive to ensure the sender controls its rate (they may also arrange for the sender to pay these charges). Assuming notifications of received marks are reflected to their source, the sender gains full knowledge of the effect of its action on everyone else sharing the same path resources (but delayed by one round trip time). Therefore the sender has complete fine-grained freedom to control the quality of its packet stream, tempered by a mechanism to encourage responsible behaviour at a matching time scale.

Happily, as well as being theoretically near-optimal, such a service plan is highly practical to deploy, as pricing can be applied to the ECN *protocol*, which is an existing standards track Internet technology.[15] The default *algorithms* in that standard assume the sender will confine itself to a standard behaviour, rather than giving it full freedom. But, during the standards process, we ensured that the addition of the SPSP responsibility mechanisms and free choice of control algorithm is possible within the framework of that same standard [49]. This is achieved by partitioning traffic into two diffserv classes, one to be used with the standardised 'buffer filling' algorithms, the other prioritised over the first and used for an SPSP-like low latency 'buffer starving' service [21].

We have extensively studied the feasibility of service plans like SPSP in the M3I project, but for the remainder of this discussion we will merely treat SPSP as the extreme, yet concrete, example of customer choice — self-admission control [29].

To summarise so far, we wish to give network customers coarse-grained choice between having or not having

[14] Or other policy control, but in our context of a mass market, pricing is the most practical means to encourage responsible usage.

[15] Thus satisfying our principle that pricing should only be applied to features of a service that already have a purpose independent of pricing.

fine-grained choice. That is, a coarse-grained choice across a spectrum of service plans ranging from fine-grained choice of sending behaviour towards the other extreme, where the customer delegates increasing levels of control to the network provider. Choices between and within points on the spectrum cannot be completely free, but must be tempered to promote responsibility, e.g. by pricing. The pricing dynamics have to be able to match that of the original control.

3.1.1 Interworking

Interworking is a pre-requisite of choice — it is no use providers offering multiple choices of service plan if none of them work with each other. The question we now pose is whether we can predict any fundamental constraints on how different control architectures can be mixed in an internetwork. We only have detailed experience of designing the interworking between two control architectures (Intserv and SPSP). But from this experience, we believe we can predict at least one general interworking constraint.

Earlier we discussed how a party not in inherent control of a capability can request control through QoS signalling, but that the inherent controller has to remain in charge between signals, merely acting on the advice of the last request. If a core network takes control decisions between signals, it cannot also delegate control to an outer network on the same timescale — neither outer networks nor their customers would have any choice — they would never be able to take nor delegate control. Therefore if both customer and network control are ever to co-exist, the core should delegate fine-grained control outwards, and any fine-grained network-centric control must only be introduced at the very edge networks around the Internet.

This line of argument needs further development, in particular, we need to consider what 'outward' means, with respect to the inherent direction of control that different QoS technologies exhibit. But an interim conclusion can be drawn that only per packet QoS technologies should be used in core networks, if heterogeneity is ever to be allowed. That is, ECN and/or DSCP but not Diffserv SLAs/policing and not Intserv reservations in the core. Happily, these per-packet technologies have also been designed to be the most suitable for core networks, being the simplest, most distributed and most scalable.

So we argue strongly that the whole network architecture should be built assuming that *some* end system customers will want full control. But we certainly do not expect full customer control to be a *common* requirement *in the short term*. In the longer term, however, we must assume there will be a far greater proportion of unattended traffic between sets of computers, where nano-second control and pricing decisions will be common. Today's network architecture must not preclude that future.

3.2 Customer tensions

As we have said, where we have requirements in tension, we cannot just give customers choice between the requirements as a glib panacea; choice brings its own problems, best understood if one imagines a human unaided by a computer being subjected to the raw SPSP service plan. We categorise these new problems into:

Excessive intervention — choice vs. convenience Having to evaluate the value of offered service and make buying decisions far too often compared to the benefit derived.

Excessive dynamics — choice vs. predictability Not being assured of a consistent enough deal on service value for a long enough duration.

Excessive customisation — choice vs. transparency Not being able to effectively compare prices between competitors, because every path through the network at every time of day has a different price — no similar products from different vendors — effectively excessive customisation.

We discuss solutions to each of these problems in the following sections. In each case, the solution is synthesised at the edge of the network. Then, because these are only human-based problems, our solutions can be by-passed for applications that can cope with the raw choices from which they are synthesised.

3.2.1 Choice versus convenience

The obvious solution to too much human intervention is to get a computer to make the buying decisions, but based on a human-created policy. Such a dynamic price handling (DPH) agent is shown conceptually in Fig

7. Both network providers are shown offering the fully dynamic SPSP service plan to each other and their customers. The customer on the left runs an agent which optimises the net value gained from the network service against a QoS buying policy specific to the session. A buying policy is essentially a plot of preferred rate[16] against prevailing price, derived from the economic value that typical users derive from a similar task using focused user experiments [31].

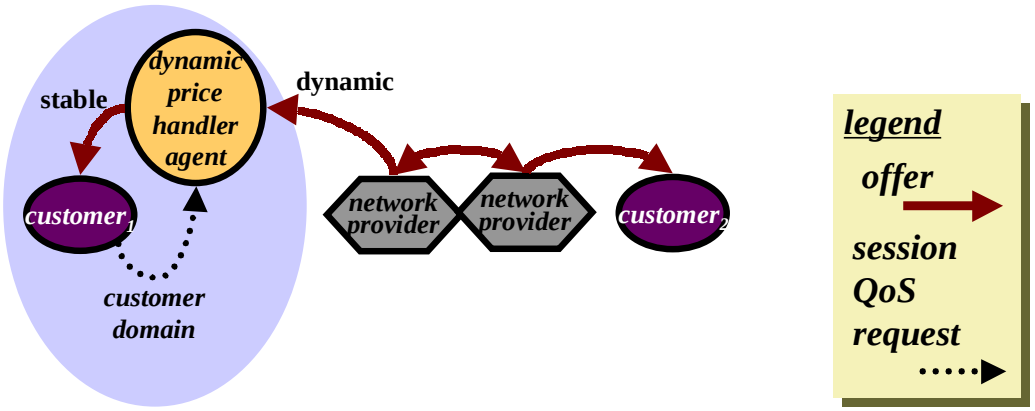

Figure 7: Dynamic price handler

A fully working dynamic price handler to intermediate between the SPSP service plan and the customer (DPH/ECN) has been designed and implemented in the M3I project [32]. The nature of buying polices, how they are matched to application tasks and the stability of various congestion control algorithms have been extensively analysed and explored [48, Section 1]. This agent clearly resolves the tension between convenience and choice; even choice on per packet time scales.

3.2.2 Choice versus predictability

Even with the SPSP service plan, a dynamic price handler can hold a rock steady bit-rate, but of course, at the expense of an unpredictable price. Similarly it can hold the price steady if it is allowed to vary the rate. Which of these behaviours results, or anything in between, is merely a matter of using the correct shape buying policy. What it cannot do, is hold *both* steady for the duration of a session. This is the consequence of a more fundamental predictability dilemma. If users all want to behave completely unpredictably on a network, they cannot expect the aggregate response of the network to be completely predictable in return (a tension of openness vs. protection or freedom vs. responsibility).

However, in a large SPSP network, as long as there is a mix of handler policies, some with elastic and some with inelastic buying policies, the inelastic ones can achieve great predictability gains over a session, because the elastic ones give way. Essentially, a dynamic price handler results in *smoothed* but not perfect predictability.

Three alternative solutions to this dilemma have been proposed and examined in the M3I project:

- The 'user direct' scenario (meaning direct user control), which is quite similar in concept to Odlyzko's Paris Metro Pricing [46]. Here strictly prioritised differentiated services classes are provided, but with no service level agreement, nor policing nor re-marking. Use of each is priced per volume on a rising scale against rising class. When the user perceives QoS is degrading in one class, she can switch to a higher class and vice versa. But the price of any one class is constant over a long time, and hence pricing is completely predictable with changes in charge under user control. Of course, QoS is variable depending on congestion, so the dilemma has not been completely solved; ease of maintaining price stability has merely been prioritised over ease of maintaining QoS stability.

[16] Other dimensions of QoS than rate can be included, but are not relevant for SPSP which keeps latency close to the propagation delay minimum for all paths.

- The clearinghouse introduced earlier (Section 2.1.2) — this simply moves the problem of price stability to the customer(s) at the other end of a communication and is therefore only applicable if one customer is less averse to price instability than the other.
- The 'guaranteed stream provider' (GSP) is shown in Fig 8. It is similar in concept to the 'gateway' proposed by Gibbens & Kelly [30]. This truly does solve the predictability dilemma by introducing another party to broker the risk of price variations, thus synthesising price and QoS stability at the edge of the network. It is discussed further below.

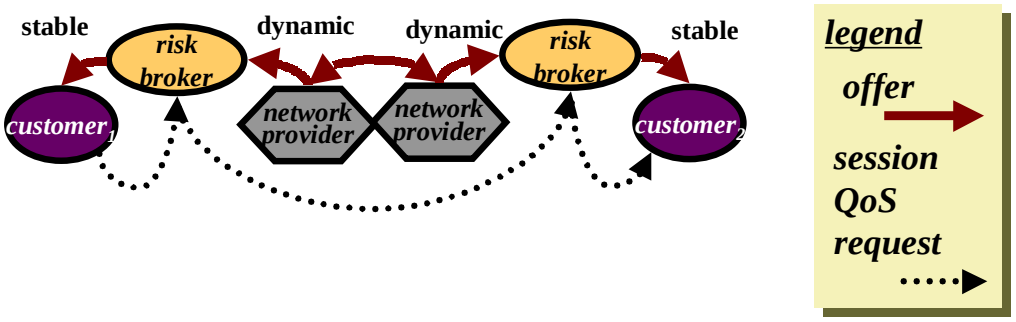

Figure 8: Guaranteed stream provider

Each guaranteed stream provider gateway consists of two functions: the financial function of a risk broker and the technical function of a gateway between QoS mechanisms (RSVP at the edge and ECN in the core). Intserv-like reservations can be made by customers, and are offered by the provider at a fixed price. The guaranteed stream provider arranges these special gateways all around its network, and reservation signalling crosses the network from end to end, but is ignored by internal routers, which are not RSVP-enabled. The routers between the gateways use congestion marking. If they are operated by a separate provider, SPSP pricing can be used. The receiver's gateway applies call admission control to reservation requests, informed by the current level of congestion seen between it and the other gateway on the path required for the reservation. This allows the gateway to aggregate congestion information from many flows, so that reservation requests can usually be processed immediately, as congestion levels on the path can be known in advance of any reservation request.

This gateway has been implemented and a number of variations have been studied in great depth in the M3I project [34, 4, 35][48, Sections 3 & 5]. In one of the most interesting variations, a network with no reservation capabilities except round the edges can synthesise true QoS guarantees across all paths. It is a very practical way to synthesise a connection-oriented service familiar to the retail market out of the rather unfamiliar SPSP recommended in the core.

The guaranteed stream provider solves the convenience and transparency dilemmas too. Its only issue is that a reservation-based service precludes a source from being unpredictable and is therefore only a suitable QoS service for streaming applications and the like (predictability vs. openness to evolution of new application types). This dilemma can only be solved by combining with other, more dynamic and open schemes. However, the important advantage is that the reservation capability is only synthesised at the edge, so it is easy for more ambitious QoS schemes to bypass it and gain direct access to the more flexible service plan behind it.

3.2.3 Choice versus transparency

As we have said, SPSP gives the extreme of choice, creating different pricing for every path, pricing therefore becoming difficult to compare across providers. The dynamic price handler agent does not and cannot solve this problem, while all the other solutions discussed above do, because they reduce control and choice to manageable proportions. The guaranteed stream provider is particularly interesting in this respect because it converts the extra probability of congestion on certain paths into higher probability of call-blocking, thus hiding the extra price complexity and solving the price transparency dilemma.

However, in a sufficiently competitive market, price transparency can be assured over the most popular routes.

If a provider gets away with adding spoof congestion signalling to make excess profits on other routes with less competition, it is surely quite entitled to harvest the returns if no other provider is smart enough to compete for that route.

We discuss provider tensions like these in the section following the next, after a brief summary of the solutions we have presented to our customer tensions.

3.2.4 Customer tensions: Summary

At the end of Section 2.3.3 on QoS technologies, we had reached the point where we had identified the fundamental constraints that QoS signalling technologies place on QoS stability (minimum refusal time scale) and price stability (minimum time scale of any type of QoS signal). This allows us to place the two most well-known service plans, Diffserv and Intserv, in the QoS-price stability space shown in Fig 9. We have also placed the sample path shadow pricing (SPSP) service plan in the same context, but a 'raw' version, that is one without any sensible policy controlled agent (a raw SPSP has never been proposed by anyone — it is just a hypothetical concept reacting randomly, rather than with any policy).

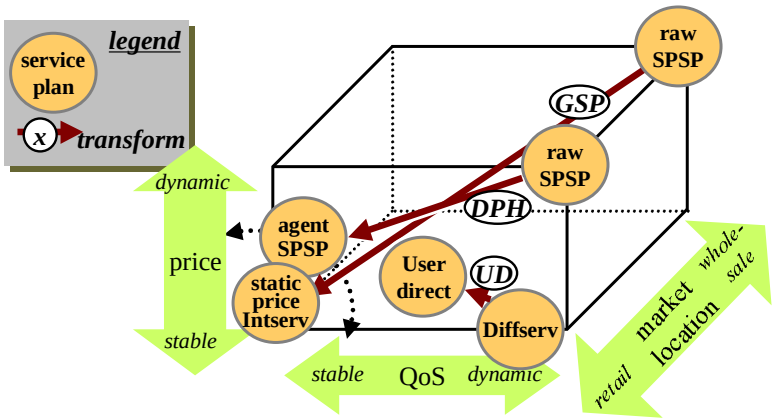

Figure 9: M3I scenarios
UD: User Direct; GSP: Guaranteed Stream Provider; DPH: Dynamic Price Handler

We have then added transformations to the same diagram that represent each of the above functions:

The dynamic price handler transforms very dynamic raw SPSP into near-Intserv QoS stability or near-Intserv price stability (not both at the same time). The result may sit anywhere on the dotted arrow, depending on the policy used;

User direct; Direct user intervention transforms the inability of Diffserv to adapt to congestion into slightly more adaptive QoS, at the expense of having to accept a controlled degree of price variation;

The guaranteed stream provider transforms raw SPSP into completely stable Intserv, with respect to price and QoS. Note that a third dimension has been introduced for this case, to show that QoS and price stability can only be achieved simultaneously by introducing another link in the value chain.

Table 1 summarises the information given in the preceding sections on solutions to customer tensions — the labels on the rows are short-hand for each of the issues raised and should not be interpreted emotively! There is no implication that all the features have equal value or that the 'best' solution is the one with most features — other unmentioned factors always need to be taken into account. For instance, the guaranteed stream provider appears 'best' but it involves more network infrastructure than those to its right and only works with gateways at both ends of each path.

The main message from the experience reported in these sections is that it is practical to synthesise useful combinations of features around the edges of a network that emulate other QoS technologies, but only using absolutely minimal technology in the core — **minimise then synthesise**.

	Intserv	Diffserv	GSP	UD	DPH
openness	□	□	□	■	■
price predictability	■	■	■	▼	▼
QoS predictability	■	□	■	△	▲
convenience	■	■	■	□	■
price transparency	■	■	■	■	□

Table 1: Solutions to tensions between requirements
Legend for presence of feature:
■: yes; □: no; ▼▲: mutually exclusive; ▼ △: biased mutually exclusive

4 M3I Architecture: overview

So far, we have discussed principles with a great deal of abstraction. It is probably difficult to envisage how the principles above might be applied during the design process of a network and its supporting systems. Part II of this document [20] describes the construction of the architecture in detail, but here we give an overview, using the opportunity to link back to the principles wherever relevant, in order to make them concrete.

4.1 What is M3I technology?

We have discussed how the M3I Architecture allows different providers to deploy their own particular flavour of service plan. But this doesn't answer what it actually is that providers deploy, nor what they deploy to. The answer is threefold:

Operations IP network middleware on customer systems giving their buying policies real-time control over application quality
and IP network middleware on provider systems giving their selling policies real-time control over network quality

Configuration a framework in which the middleware sits to enable switching between service plans

Transformation middleware between customer and provider to transform service plans along the value chain

Fourthly, M3I technology is an *approach* to managing network resources using pricing, even if hidden from customers (shadow pricing). The approach is grounded in the principles laid out in the previous section, further complemented by the considerable body of work in Songhurst *et al* [52]. Below we give an overview of each of the first three aspects.

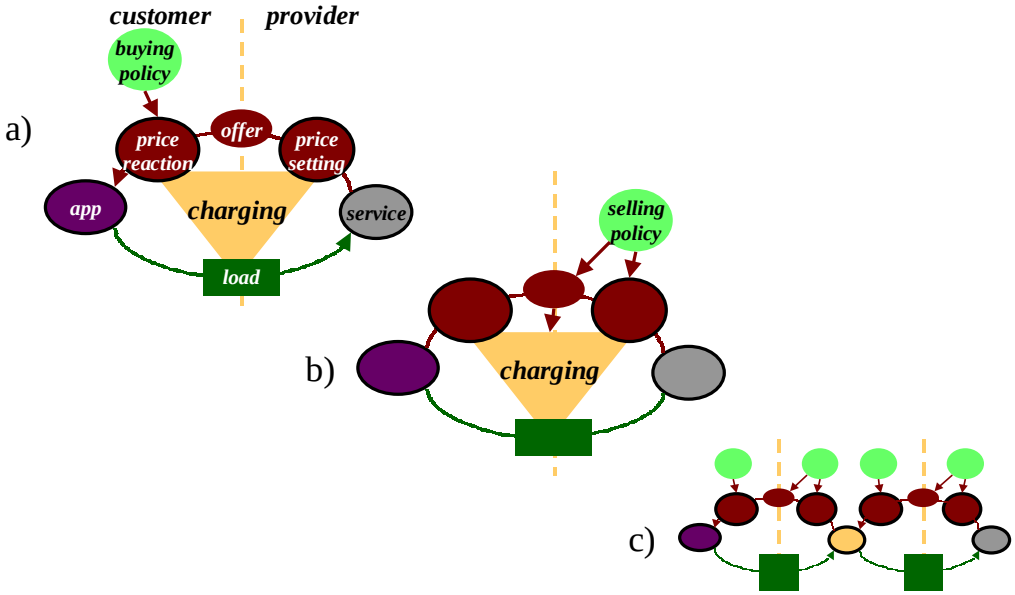

Figure 10: What M3I technology is:
a) Operational price control within a service plan
b) Configuration of a service plan
c) Gateways between service plans

Operational price control within a service plan (Fig 10a). We have chosen to always represent the architecture as a customer-provider interface, along with the components that use the interface in each

party on either side. The idea is that the same architecture can be used at each provider-provider interface throughout the Internet. The split-edge pricing principle allows an edge provider to encapsulate all the onward wholesale interfaces.

The main components lie in a control loop. The load offered to the network by the customer's application is shown as the lower half of the loop, which is in turn controlled by the 'market control plane' components shown running over the top, from network back to application. The **offer** is the primary object that the provider uses to describe its approach to the market, as discussed in earlier sections. The offer determines how frequently control signalling is required (an inner control loop, not shown). The offer is an information structure created centrally by the provider's business processes and placed in an offer directory. From here it can be transported with a suitable protocol, to be shared with customers and potential customers, who may cache it on their own systems. There is no implication that signalling is flying across the market control plane as frequently as data is flying across the service interface below. The offer *can* be quickly fine-tuned by altering relative price weightings within it, either manually, or using the price setting component, but a good service plan should be able to regulate demand at this level without extremely frequent interference.

Similarly, the customer doesn't want to be hassled with reacting to regular price and tariff changes, which *can* be dealt with by the **price reaction** component, either silently, or in an advisory mode. The price reaction component is installed as middleware on customer machines, either distributed, or centrally for a small number of machines.[17]

Each task that the customer undertakes may require a different price reaction strategy. So the customer uses **buying policies** as the primary object to control the price reaction component. Like offers, buying policies are information structures. The intention is that buying policies are set by the party(ies) paying for an end to end application, and may be transported to other participants using a suitable session-control protocol. Usually default policies, built-in to an application or operating system, will be matched to the task in hand, to reduce user intervention. However, it is always possible for a user to tweak her policy away from the default.

The **charging and accounting system**, shown in the middle of the loop, polices the market for the provider, and ensures the price setting and the price reaction components have sufficient charge feedback to be able to make control decisions. With some service plans, some charging and accounting functions are best distributed to the customer for performance (recall the earlier advice about bulk charging with the option of fine itemisation delegated to the customer). Hence charging is shown shared across both provider and customer as a general case, though it is often entirely provider-centric.

To summarise, two commerce-related interfaces are added to the original network service interface at the customer-provider boundary: the offer interface and the charging & accounting interface.

Configuration of a service plan (Fig 10b). This is the point at which the provider chooses its approach to the market. This will be instantiated as the public offer, as already described. But there will also be a private policy behind the offer, containing the strategy on how to develop the offer over time, or how to vary pricing in response to competition or demand. The structure of the offer determines the configuration of the charging system — policy-based charging and accounting.

The customer has an equivalent configuration phase (not shown) where she assesses the offers available on the market, and accepts one for the task in hand (or longer). The accepted offer then configures her price reaction and charging components.

These configuration activities (which also include creation and destruction) have not been the focus of the M3I project investigations, but they are the strong architectural context for the other two aspects of the work.

Gateways between service plans (Fig 10c). Using **money** as a common control currency has allowed us to envisage a market control system across providers who are each using completely different approaches. When we discussed split-edge pricing, we explained that multiple standards could co-exist in the market if certain network providers were to buy in service under one service plan, but sell it on a completely different plan. Such providers require an extra component to do the transformation of control, which is shown between the incoming price reaction and outgoing price setting functions of the middle provider in the figure. Whether

[17] The name of this component is strictly incorrect, but became stuck early in the M3I project. It actually assesses net value, so it reacts to changes in how the user assesses value and to changes in charge unrelated to price changes, as much as to straightforward price changes.

this function is distributed across routing equipment or centralised per provider is impossible to say at this general, high level, as it depends very much on what transformation is being attempted.

It is expected that arbitrary non-money-based policies will also be applied to control QoS (e.g. for corporate networks), but we assume such transformations will be the last in the chain before the end user, as they will tend to use local concepts like department names, application types, etc. We do not believe such arbitrary policies will ever have to be transformed back into money-based policies, which would present intractable problems.

Each provider's service plan is described as an architecture by those that have thought that the industry should standardise around one plan (e.g. Intserv or Diffserv). Therefore, the M3I Architecture can be thought of as an architecture of architectures. This sounds very grand, but we should warn that we have no general solution to interworking between architectures. We do offer the principle of split-edge pricing and of bipartisan contracts to ensure there is a way for new 'standards' to start and spread. We also offer the principle that core networks should never use QoS technologies that take control across timescales greater than the packet, otherwise choice of control architectures will never be possible. But, although we have solved a number of transformations at the *tariff* level, we have only set out to solve specific *QoS technology* interworking problems in one case. Although this work was impressively successful, this does not imply there will be interworking solutions between every combination of technologies.

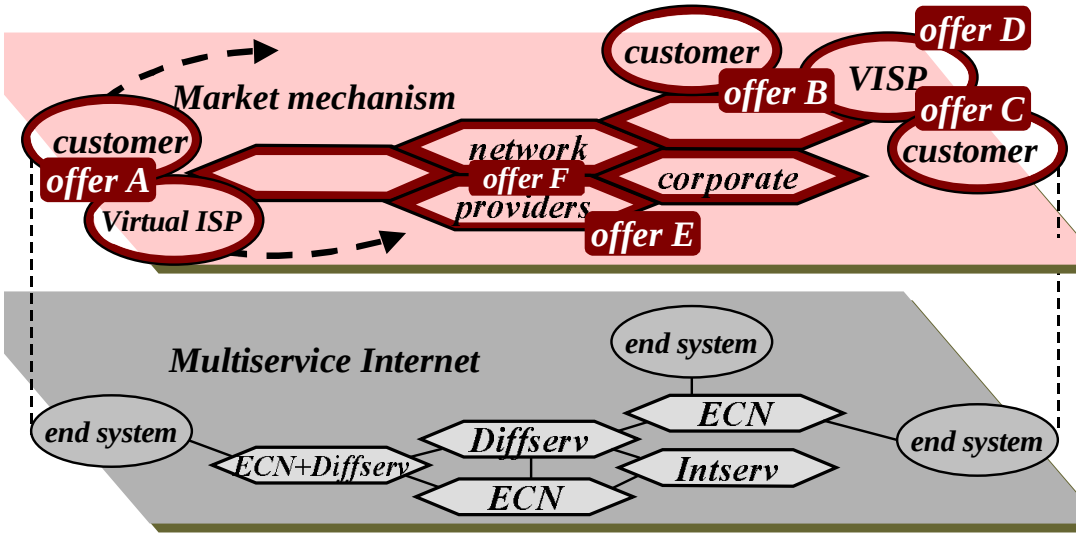

Figure 11: Market control plane

An internetwork of market controlled networks Fig 11 shows the result of putting together all the above aspects. The technical internetwork is depicted in the lower plane labelled 'Multi-service Internet'. It is shown using different combinations of QoS technology in each network. Above it are the market mechanisms depicted as thin wrappers around each network and around each customer. These form the business interfaces between them (imagine looking down from above, so that the technical network beneath is visible within the white holes in each wrapper). These 'market mechanisms' and 'business interfaces' in this 'thin wrapper' are those described above: price setting, offer and price reaction (Fig 10a).

An exemplary selection of the available offers are shown being presented at the edges of some provider networks. The network making offer B is selling service both directly to a customer and to a virtual Internet Service Provider (VISP). In turn the VISP is re-selling the service to another customer via offer C. Of course a VISP, or any network provider can form more than one offer for its service (e.g. offers C & D). Note the VISP does not map down to any technical network infrastructure, being a purely commercial intermediary. The dashed arrow from the customer on the left shows that this customer could choose another offer than A, either because it is mobile, or because competing offers are available to it without moving. The same is true for virtual ISPs, who

can re-sell the service of many network providers, or switch between them.

Also note that the same market interfaces are present between providers (e.g. offer F) and for customers who are large operators themselves (e.g. the corporate network bottom right). The detail of price setting by providers, reaction by customers, accounting etc., that sits in these 'thin wrappers' is given in the use-cases in Part II of this document.

4.2 Sub-systems

The previous section introduced the major M3I Technology components in the context of the life cycle of a single network's service plan — operation and configuration — and then interworking with other networks. To aid understanding, the same components can be classified into sub-systems, which are *loosely* layered. Fig 12 shows the four major subsystems:

- **network service**
- **market control** of its load
- **application and policy control** of the market (by **human customers or providers**)
- and **charging** within all this.

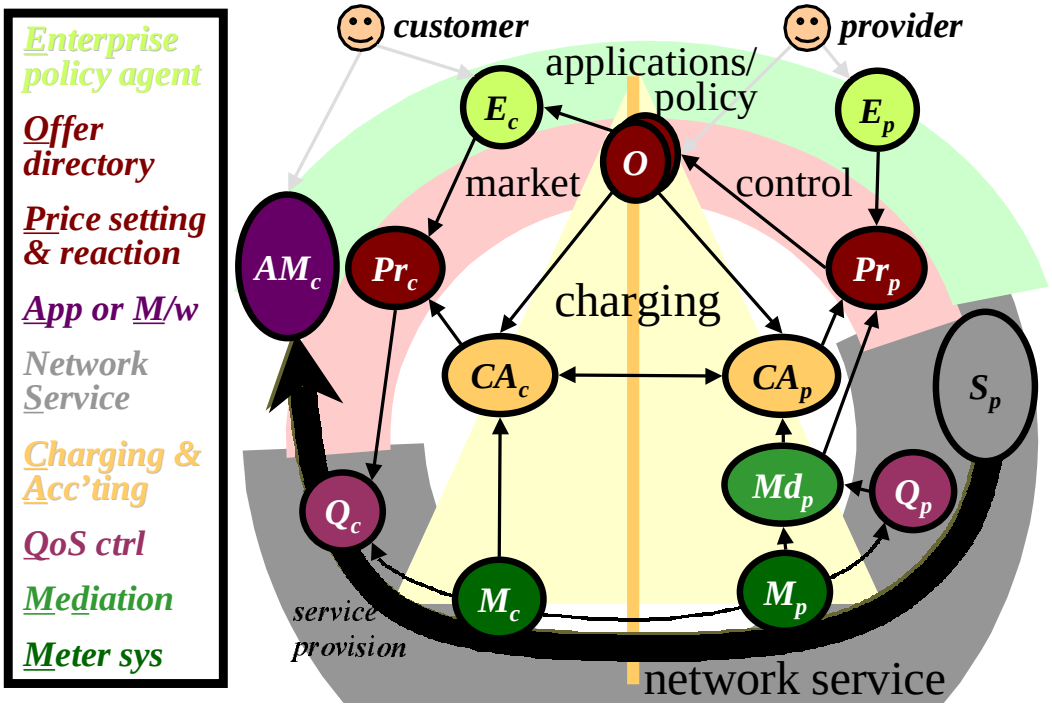

Figure 12: High level architecture

It should be possible to recognise the general customer-provider layout and three main interfaces between them of Fig 10a) described earlier, but with more detail and layered sub-systems shown as background shading. Note that we are discussing high level *control* plane layering, so there is no implication that all *data* is processed by passing through these layers. In fact, the data path is shown by the thick arrow in the direction of delivery of the network service, from provider to customer (irrespective of the fact that data is flowing in both directions, or that the offer may price each direction differently). The inter-network use-case in Part II explains how essentially the same architecture is applicable at the interface between two peer providers, because in reality their relationship is the overlay of their mutual customer-provider and provider-customer statuses.

It will become clear in Part II of this document that each different scenario chosen by a provider will require slightly different arrangements of components talking with each other. The diagram is the most complete one

we could draw, including every possible aspect of a market managed multi-service Internet infrastructure. It will be seen that different service plans omit different parts of this general architecture, requiring some rewiring of the communications paths. Nonetheless, this rewiring stays within the layering, which expects components in one layer to talk with each other and with those components above and below. We term it loose layering, not because it can be broken, but because the application is allowed to implement the function of any layer if it chooses not to delegate its control to lower layers [25]. Also charging, accounting and metering are not really a layer — their service is orthogonal to the rest of the system, but without it there would be no bite to the incentives of the market mechanisms.

4.3 Architectural approach

Here we give a brief overview of the approach taken in Part II [20] of this document so that the present part (Part I) may stand on its own if required.

The technique is to identify a small but sufficient set of **service building blocks** from which we can compose all four of the above major sub-systems. The definition of each building block includes its function and the type of its input and output interfaces. This leads to a small set of **interface types** that can be used to connect together any **composition**. Altogether there are fourteen service building blocks and five families of interface types. Most interface types are sub-typed to define more specific interfaces, but whole families are also dealt with together in some circumstances, motivating their high level classifications.

The distinction between **operation** and **configuration** is maintained throughout the architecture. Configuration itself occurs at two further distinct levels of granularity: configuration of a network **service** itself then configuration of **application services and of sessions** that use the network service.

Configuration of application sessions is discussed in the next section, to allow us to emphasise how the architecture offers an extremely open creation environment, encouraging the invention and exploitation of novel application business models, incorporating the business models of network providers.

4.4 Applications overview

We now draw a line between what is inside and what is outside the market managed multi-service Internet infrastructure. The intention is to ensure that what is outside can be as commercially innovative as possible — extreme commercial openness. Some examples will help at this point to show how extreme the openness of such a service could become:

1. The first degree of openness allows a user to just start using a network service they have never used before *without gaining any special permissions*, as long as they have earlier contracted to pay for whatever they use. So if they install some software that uses QoS reservations, they just start getting charged for the reservations they make.
2. The second degree of openness allows anyone to buy and then re-sell the network service in novel ways on behalf of other users. For instance, a company could operate a call centre which pays for all the network usage of both itself and its customers, *without having to arrange anything* with the network provider.
3. The third degree of openness allows anyone to create software that allows anyone else to re-sell network services in novel ways. For instance, a company could sell call-centre software, that allows anyone else to operate a call centre offering service to customers across any number of (open) networks. It might even allow each call centre operator to choose from a selection of business models, which bundle the network service in different ways.

A provider's view may be that these forms of openness are arbitrage, which is to be avoided at all costs. Another is that such activity creates new business for the network, at least as a commodity. The trick is to build an architecture which allows such extreme openness, but then to be able to close it off while it is not commercially advantageous. As networking commoditises in the future, the policy can 'simply' be changed, enabling more openness and growing the market.

In order to encourage this openness, there are essentially three activities we have to support to drive a market managed network:

- selling
- buying
- using

The applications that support these three activities are defined as part of the overall architecture, but outside the infrastructure we need to build. For future flexibility, the architecture does not contain any building blocks to implement these functions, but, in order to be useful, it clearly must provide interfaces to support them. That is, management interfaces for buying and selling, and, of course, the service interface for actually using the network. Thus, in the exemplary use cases in Part II:

- The **selling** activity (deciding what to sell and which tariffs and price setting policies) is primarily **manual**, but the architecture legitimately supports it with directory interfaces for holding offers and provider enterprise policies.
- The **buying** activity involves the **customer enterprise policy agent**. This could be arbitrarily complex, and therefore also sits outside the architecture. It is nonetheless supported with interfaces for offering service to it and receiving offer acceptances from it. A directory interface is provided for it to create, modify and safely store its policies and the price reaction function offers an interface expecting these policies as inputs.
- The **customer's applications or middleware** are the entities **using** the network service and hence are also outside the infrastructure. Obviously, the service interface of the network is provided for their use.

However there is still room to take a step yet further outward. Instead of trying to offer a suitable service for what people believe they should be able to do now, let us consider what is necessary to encourage people to do *anything* with the network — total commercial openness. That is, to support people who wish to offer their own idea of the network service bundled into their own services. Thus, as well as considering how to service current needs, we must serve the creation of new needs. From this perspective it becomes important to offer management interfaces for two further classes of application:

- selling and buying services that use the network service
- selling and buying sessions that use the network service

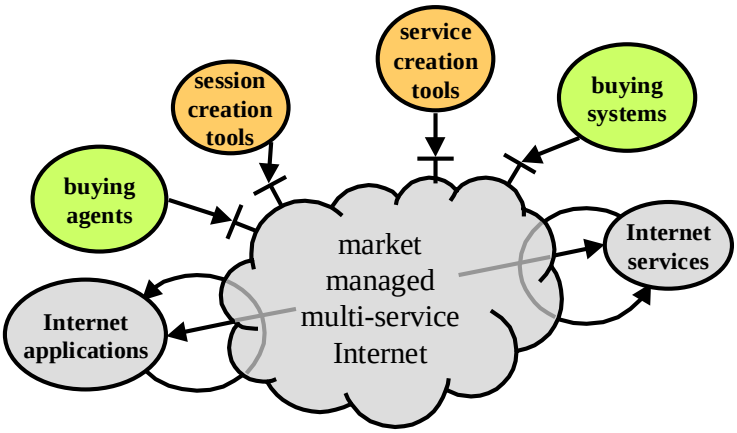

Figure 13: Applications of a M3I

Thus, our building blocks will offer interfaces to support creation of complementary services and sessions, not just network service itself (see Fig 13). We will further justify these choices in a later section where we define generic building blocks to represent high level classes of characteristic applications (see the Applications section of Part II [20]).

5 Conclusions

5.1 Limitations & further work

An architecture not only helps the mind form higher level concepts, it invariably limits the mind to exclude concepts that may be valid, but don't fit the architecture.

One of our over-arching assumptions is that the world's networks will be unavoidably controlled by a global market. We are assuming that polices that make judgements based on the identities of corporate bodies or on concepts like social worth will not be globally meaningful, unlike the more universally understood supply, demand and price. We assume policies with richer semantics will be applied locally, rather than with more global reach, but more work is needed to understand how the two inter-relate.

5.2 Summary

The architectural work in the Market Managed Multi-service Internet (M3I) project has produced a control architecture for the Internet that enables and manages network supply and demand over a spectrum of time scales from sub-milliseconds to years. It encompasses Internet control architectures like Intserv and Diffserv as well others, including a selection proposed within the M3I project. The Architecture is designed to allow competing approaches to co-exist across the Internet. It takes the form of both concrete things like building blocks and interfaces (Part II [20]) as well as principles and recommendations to be used in their composition and configuration (Part I — this part).

Status The kit of parts and particularly the principles on how to put them together result from analysis of fundamental issues, being designed to assure the evolvability of the Internet over decades. Although such ambitious research can always be considered 'for discussion' and therefore 'work in progress', this document is the culmination of a body of work over a number of years, which we now believe is complete enough and stable enough to be used decisively.

Mechanisms for choice of control architecture We have identified that the important building blocks of any specific control architecture are the underlying standard QoS signalling protocols (RSVP, DSCP, ECN etc.) and, perhaps surprisingly, the **tariff**. A provider's tariff gives each protocol its significance, in effect describing the choice of control architecture. Thus, the tariff *is* the primary policy mechanism for introducing new control architectures, including controlling re-configuration of dependent systems. Strictly, the relevant policy object is the whole service offer of which the tariff is just a part, but it is the tariff that has most significance, as you generally get nothing unless you pay for it.

Do we need a choice of control architecture? QoS for the Internet has been a long-running problem, with many proposals for solutions. Two have gained particular prominence: Intserv and more recently Diffserv. Since about 1997, a more radical set of approaches have been gaining ground, which give full control to the customer e.g. sample path shadow pricing (SPSP). If these proposals were just different ways of doing the same thing, we would simply recommend choosing one and rejecting the others. However, they are fundamentally different: in the guarantees they can offer; in where the cost of complexity lies; and in their potential to meet innovative requirements that might arise in the future. Because of these fundamental differences, choice of QoS architecture will be the major differentiator between network providers, along with pricing of course.

Open by design, closed by choice Through thorough investigation, we have found fine-grained customer control of QoS has considerable potential. But, it would be highly controversial to push all fine-grained control outwards to the networks' customers, however essential for future evolution potential this might turn out to be (end to end principle). We recognise that network providers and their suppliers see quality of service as their main added value and will not want to do this. The strength of our approach is to bring the politics of who controls QoS under **policy** control, rather than embedding today's choices in tomorrow's technology. Our main approach is traditional; to decompose the problem into minimal functions, then synthesise

desired control architectures from the parts — '**minimise then synthesise**'. Also unsurprisingly, we take care to separate policy from function. The difference is that we recommend, in effect, pushing control out to the customer, then grasping it back, under the policy control of tariffs. That is, designing the network as if it were customer controlled, then re-synthesising network control around the edge networks. 'Open by design, closed by choice' puts providers in control of when and if they give up control.

Customer controlled QoS In order for evolution to customer controlled QoS architectures to be worthwhile, we have to show they are feasible and that they could open up significantly new markets. All QoS proposals involve some customer control, differing only in how often they allow it (sub-milliseconds to months). Fine-grained customer control of QoS has been investigated in depth in the M3I project. Although there is still research to do, we have shown it is possible to synthesise QoS guarantees at the edge of the network that are indistinguishable from guarantees given by network-embedded technology. Indeed, as wireless access predominates, we predict that end system QoS adaptation will become ubiquitous, making network QoS guarantees redundant in most cases. Also network-based control is less flexible for the increasing proportion of traffic with no direct human involvement. We can start to see that the long term potential of customer control could eventually outweigh any short term gains from embedding control in the network.

Industry structure for evolution of choice In order for there to be a choice between control architectures, new ones have to be able to be introduced alongside the old. We have shown how the principles of **edge pricing** and **bipartisan contracts** must be strictly adhered to, for this to be possible. We have shown how some popular service models like mobile roaming break these principles and how to redesign them correctly to achieve the same result. We have also analysed what might be called the **politics of control** down to sub-millisecond time scales (nano-politics?!). We have shown how the freedom to choose fine-grained customer control depends on network providers not taking coarser-grained decisions first. The consequence is that coarse-grained control architectures should be confined to networks around the edge only, with only per packet QoS technologies closer to the core. Then, if evolution to customer control is desired, the edge architectures can be bypassed to expose the raw capabilities behind them. This is important, as both Diffserv policing and Intserv reservations are coarse-grained, and should therefore not be deployed in the core (note that we make a distinction between Diffserv policing and diffserv per hop behaviours, which are fine in the core). Usually such decisions are based on technical merit arguments, such as scalability, but we believe it is just as valid to conduct this argument in the context of control politics.

Too much choice? By investigating scenarios that offer extreme levels of raw customer control, we have found that convenience, predictability and transparency, with respect to both QoS and price, suffer. With our technique of synthesising different control architectures at the edge, and keeping to all our other principles, we have managed to create service plans that solve most of these problems, and one solution that solves them all at once. But there remains a fundamental **tension**: between closed solutions where the provider solves these problems or open solutions where the customer is left to solve them, but has far more freedom to use the network in novel ways. The closed solution is more suitable where humans are the direct users of the network. The open solution is more suitable where 'software is the customer'. This software insulates the customer from the excessive choice, absorbing the inconvenience, unpredictability etc. and deciding its priorities within its own logic. Opening up the raw interface to QoS (that sits behind the one for mere humans) will bring the full potential of the computing industry to bear — to exploit the full potential of the multi-service Internet. But where 'multi-service' doesn't only mean multiple services are *provided*, but also multiple services can be *synthesised*, by anyone else.

M3I Architecture The M3I Architecture has been designed to realise the above choices. It is a control plane for the Internet, that gives control of the Internet to the market in communications, at whatever granularity is chosen through policy. And by policy we deliberately mean the combined policy of providers who choose what to offer, and customers who decide the winners.

The M3I Architecture specifies a set of common components that sit either side of any customer-provider interface on the Internet. It adds two market-related interfaces to the raw network service interface: one for contractual offers and one for accounting and charging. It is the contractual offer that determines the choice of control architecture, as described above. The M3I Architecture is effectively an architecture of architectures, although we use the term 'service plan' (tariff/QoS technology pair) for each sub-architecture.

The Architecture covers the configuration of new service plans and switching between them, although these have only been theoretical exercises, not being the focus of practical work in the project. The M3I Architecture has mainly been exercised to design and build a number of individual service plans, synthesised out of its building blocks and interfaces. It has also been used to build an operational gateway transforming between two service plans.

Acknowledgements

Many contributors to the discussions in the M3I consortium, but especially Martin Karsten and Oliver Heckman (TU Darmstadt), Ragnar Andreassen (Telenor), Burkhard Stiller (ETH Zürich), Panayotis Antoniadis and Costas Courcoubetis (AUEB), Mike Rizzo, Jérôme Tassel, Kostas Damianakis, Steve Rudkin, Ben Strulo, Trevor Burbridge, Kennedy Cheng, David Songhurst, Arnaud Jacquet, Clazien Wezeman and Gabriele Corliano (BT), Frank Kelly, Richard Gibbens and Jon Crowcroft (Cam Uni).

References

[1] Jörn Altmann. M3I; ISP business model report; prototype descriptions. Deliverable 7.2, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, January 2001.

[2] Jörn Altmann, Hans Daanen, Huw Oliver, and Alfonso Sánchez-Beato Suárez. How to market-manage a QoS network. In *Proc. IEEE Conference on Computer Communications (Infocom'02)*, June 2002.

[3] Ragnar Andreassen (Ed.). M3I; Requirements specifications; reference model. Deliverable 1, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/, July 2000.

[4] Panayotis Antoniadis (Ed.). M3I; the guaranteed stream provider. Technical Report ?, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, February 2002.

[5] Arbinet Web site. Web page URL: http://www.arbinet.com/, 1999+.

[6] F. Baker, B. Braden, S. Bradner, A. Mankin, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang. Resource ReSerVation protocol (RSVP) — version 1 applicability statement; Some guidelines on deployment. Request for comments 2208, Internet Engineering Task Force, URL: rfc2208.txt, January 1997.

[7] Band-X Web site. Web page URL: http://www.band-x.com/, 1999+.

[8] L. Berger and T. O'Malley. RSVP extensions for IPSEC data flows. Request for comments 2207, Internet Engineering Task Force, URL: rfc2207.txt, September 1997.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Request for comments 2475, Internet Engineering Task Force, URL: rfc2475.txt, December 1998.

[10] Anna Bouch. *A Users' Perspective of Network QoS and Charging*. PhD thesis, UC London, URL: http://www.cs.ucl.ac.uk/staff/A.Bouch/pub.html, 2001.

[11] Anna Bouch and M. Angela Sasse. Network quality of service: What do users need? In *Proc. 4th International Distributed Conference. Madrid*, URL: http://www.cs.ucl.ac.uk/staff/A.Bouch/pub.html, September 1999.

[12] Anna Bouch and M. Angela Sasse. The case for predictable network service. In K. Nahrstedt and W. Feng, editors, *Proc. MMCN'2000. San Jose, CA*, pages 188–195, URL: http://www.cs.ucl.ac.uk/staff/A.Sasse/pub.html, January 2000.

[13] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: an overview. Request for comments 1633, Internet Engineering Task Force, URL: rfc1633.txt, June 1994.

[14] Robert Braden, David Clark, Scott Shenker, and John Wroclawski. Developing a next-generation Internet architecture. White paper, DARPA, URL: http://www.isi.edu/newarch/DOCUMENTS/WhitePaper.pdf, July 2000.

[15] R. Braden (Ed.), L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) — version 1 functional specification. Request for comments 2205, Internet Engineering Task Force, URL: rfc2205.txt, September 1997.

[16] Lee Breslau, Edward W. Knightly, Scott Shenker, Ian Stoica, and Hui Zhang. Endpoint admission control: Architectural issues and performance. *Proc. ACM SIGCOMM'00, Computer Communication Review*, 30(4), October 2000.

[17] Bob Briscoe. The direction of value flow in connectionless networks. In *Proc. 1st International COST264 Workshop on Networked Group Communication (NGC'99)*, volume 1736, URL: http://www.btexact.com/projects/mware.htm, November 1999. Springer LNCS. (Invited paper).

[18] Bob Briscoe. The direction of value flow in multi-service connectionless networks. In *Proc. International Conference on Telecommunicatons and E-Commerce (ICTEC'99)*, URL: http://www.btexact.com/projects/mware.htm, October 1999.

[19] Bob Briscoe. M3I; Standards activity. Deliverable 5.5, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, September 2001.

[20] Bob Briscoe. M3I Architecture PtII: Construction. Deliverable 2 PtII, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, February 2002.

[21] Bob Briscoe and Jon Crowcroft. An open ECN service in the IP layer. Internet draft, Internet Engineering Task Force, URL: http://www.m3i.org/papers/draft-ietf-tsvwg-ecn-ip-00.txt, February 2001. (Expired).

[22] Bob Briscoe, Mike Rizzo, Jérôme Tassel, and Konstantinos Damianakis. Lightweight, end to end, usage-based charging for packet networks. In *Proc. IEEE Openarch 2000*, pages 77–87, URL: http://more.btexact.com/projects/mware.htm, March 2000.

[23] David Clark and Marjory Blumenthal. Rethinking the design of the Internet: The end-to-end arguments vs. the brave new world. In *Proc. Telecommunications Policy Research Conference (TPRC'00)*, URL: http://www.tprc.org/abstracts00/rethinking.pdf, September 2000.

[24] David D. Clark. A model for cost allocation and pricing in the Internet. In *Proc. MIT Workshop on Internet Economics*, URL: http://www.press.umich.edu/jep/works/ClarkModel.html, March 1995.

[25] David D. Clark and David L. Tennenhouse. Architectural considerations for a new generation of protocols. *Proc. ACM SIGCOMM'90, Computer Communication Review*, 20(4):200–208, September 1990.

[26] David D. Clark and John Wroclawski. The personal router whitepaper. Technical report, MIT, URL: http://www.ana.lcs.mit.edu/anaweb/PDF/PR_whitepaper_v2.pdf, March 2000. v 2.0.

[27] C. Courcoubetis, G. D. Stamoulis, C. Manolakis, and F. P. Kelly. An intelligent agent for optimizing QoS-for-money in priced ABR connections. *Telecommunications Systems; Special Issue on Network Economics*, June 1998. To appear (Presented: ICT'98 Porto Carras, Greece).

[28] Jon Crowcroft. User route choice: Promoting real-time ISP competition, or is traffic engineering evil? Presentation at M3I Modelling Workshop URL: http://www.m3i.org/events/0106uclworkshop/m3iucl2001crowcroft.pdf, June 2001.

[29] Richard J. Gibbens and Frank P. Kelly. Distributed connection acceptance control for a connectionless network. In *Proc. International Teletraffic Congress (ITC16), Edinburgh*, pages 941–952, URL: http://www.statslab.cam.ac.uk/~frank/dcac.html, 1999.

[30] Richard J. Gibbens and Frank P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35(12):1969–1985, December 1999.

[31] David Hands (Ed.). M3I user experiment results. Deliverable 15 Pt2, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, February 2002.

[32] Arnaud Jacquet (Ed.). M3I; price reaction: Detailed design. Deliverable 12.2, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, February 2002.

[33] Martin Karsten and Jens Schmitt. Admission control based on packet marking and feedback signalling – mechanisms, implementation and experiments. Technical Report TR-KOM-2002-03, TU-Darmstadt, URL: http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/KS02-5.\%html, May 2002.

[34] Martin Karsten (Ed.). M3I; integrate pricing system and network technology. Deliverable 12.1, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, June 2001.

[35] Martin Karsten (Ed.). GSP/ECN technology & experiments. Deliverable 15.3 PtIII, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/, February 2002. (updated by [33]).

[36] Frank P. Kelly, Aman K. Maulloo, and David K. H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[37] Tom Kelly. An ECN probe-based connection acceptance control. *ACM SIGCOMM Computer Communication Review*, 31(2):14–25, July 2001.

[38] Paul Kirkby. Management and control of next generation unified carrier networks. In *Proc. IEE 16th UK Teletraffic Symposium, Nortel Networks, Harlow, UK*, URL: http://www.jungle.bt.co.uk/projects/charging/refs/SentM&C\%20of\%20NG\%20u\%nified\%20CNs\%20UKTS-16.doc, May 2000.

[39] Pascal Kurtansky. M3I cost modelling. Deliverable 16.2, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, February 2002.

[40] Jeffrey K. MacKie-Mason and Hal Varian. Pricing congestible network resources. *IEEE Journal on Selected Areas in Communications, "Advances in the Fundamentals of Networking"*, 13(7):1141–1149, 1995.

[41] Sean McCreary and kc claffy. How far does the average packet travel on the Internet? Technical report, CAIDA, URL: http://www.caida.org/Learn/ASPL/, May 1998.

[42] Lee W. McKnight and Joseph P. Bailey, editors. *Internet Economics Workshop*, volume 2 issue 1. The Journal of Electronic Publishing, URL: http://www.press.umich.edu/jep/econTOC.html, May 1996. See also print version [43].

[43] Lee W. McKnight and Joseph P. Bailey, editors. *Internet Economics*. MIT Press, URL: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=6048, July 1998. See also electronic version [42].

[44] K. Nichols, S. Blake, F.Baker, and D. Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. Request for comments 2474, Internet Engineering Task Force, URL: rfc2474.txt, December 1998.

[45] Jakob Nielsen. Metcalfe's law in reverse. Web page URL: http://www.useit.com/alertbox/990725.html, July 1999.

[46] Andrew Odlyzko. A modest proposal for preventing Internet congestion. Technical report TR 97.35.1, AT&T Research, Florham Park, New Jersey, URL: http://www.research.att.com/~trmaster/TRs/97/97.35/97.35.1.body.ps or URL: http://www.research.att.com/~amo/doc/modest.proposal.pdf, September 1997.

[47] Andrew Odlyzko. Internet charging. In *Panel of Experts at 1st International workshop on Quality of future Internet Services (QoFIS'00)*. COST263, September 2000. (Verbal agreement only).

[48] Huw Oliver (Ed.). M3I technical experiment results. Deliverable 15 Pt3, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, February 2002.

[49] K. K. Ramakrishnan, Sally Floyd, and David Black. The addition of explicit congestion notification (ECN) to IP. Request for comments 3168, Internet Engineering Task Force, URL: rfc3168.txt, September 2001.

[50] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984. An earlier version appeared in the Second International Conference on Distributed Computing Systems (April, 1981) pages 509–512.

[51] Scott Shenker, David Clark, Deborah Estrin, and Shai Herzog. Pricing in computer networks: Reshaping the research agenda. *ACM SIGCOMM Computer Communication Review*, 26(2), April 1996.

[52] David J. Songhurst (Ed.). M3I global interaction models. Deliverable 11, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/, June 2001.

[53] Burkhard Stiller (Ed.). M3I charging and accounting system (CAS) design. Deliverable 4, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/, June 2000.

[54] Burkhard Stiller (Ed.). M3I charging and accounting system (CAS) implementation. Deliverable 13, M3I Eu Vth Framework Project IST-1999-11429, URL: http://www.m3i.org/private/, July 2001.

[55] R. Yavatkar, D. Pendarakis, and R. Guerin. A framework for policy-based admission control. Request for comments 2753, Internet Engineering Task Force, URL: rfc2753.txt, January 2000.

