New Design Principles for the Internet

Alan Ford
Roke Manor Research
Romsey, Hampshire, UK
alan.ford@roke.co.uk

Philip Eardley
BT Innovate
Martlesham Heath, Ipswich, UK
philip.eardley@bt.com

Barbara van Schewick
Stanford Law School
Stanford, CA, USA
schewick@stanford.edu

Abstract— **Socio-economic aspects are not intrinsic to the current Internet architecture. Today's architecture is becoming stressed as stakeholders introduce "hacks" to try to impose their economic desires on others, leading to a "tussle" of conflicting interests. In this paper, we propose new Internet design principles that are "designed for tussle". We believe that Internet protocols and architecture that follow them will naturally integrate both technical and socio-economic aspects, and so will be able to smoothly adapt to changes in society's demands on the Internet as they occur, without requiring permanent redesign.**

I. INTRODUCTION

Design Principles are informal guidelines to help a protocol designer or network designer achieve a solution with desirable properties. The success of the Internet to date has been, in large part, due to the perspicacity of its original design principles, enabling its huge growth in scale and scope, and enabling its abilities to incorporate technological advances and to adapt to new application requirements with extraordinary speed and economy. For example, one of the original Internet Design Principles [1] is the well known "end to end principle", which helps a design achieve the beneficial properties of resilience and evolvability.

The Internet has been less successful in accommodating conflicts resulting from differing social and economic interests of its participants, e.g,, between different operators of the network infrastructure and between the users and providers of network services. The resulting "tussle[s] in cyberspace" [2], which arise increasingly often as the demands on the Internet increase, have led to stresses on the Internet where stakeholders seek to impose their economic interests on others by distorting the architecture ("hacks"). This problem has its roots predominantly in the restricted priorities of the original designers [1] rather than fundamental flaws in the design.

One specific example concerns routing and traffic engineering. The inter-domain topology of the Internet is becoming much more strongly meshed, for example because of direct peering between multiple edge providers and multihomed end-sites. Traffic engineering is about striving for economic control over the transmission path through the topology, but the main mechanism (BGP) provides rather insensitive economic control over provider-provider interactions. This lack of flexibility manifests itself technically as growth in router table sizes and churn rates, and the inability of individual operators to manage this load without harming end to end reachability.

Another example concerns resource control. TCP's job is to share the Internet's capacity amongst the users, with each TCP flow getting the same share (for the same congestion on the path and round trip time). Historically TCP has worked well, however it has proved inadequate with the rise of new classes of application. Whilst peer-to-peer is perhaps the most prominent, others such as voice and business critical VPNs are also relevant. For example, a P2P application opens many tens of TCP flows, which squeezes the bandwidth available for applications such as web browsers that open only one or a few flows. Hence some ISPs have introduced DPI (Deep Packet Inspection) boxes, which impose (or try to impose) what the ISP considers to be a reasonable balance of resources between different users and between different applications; "reasonable balance" means, in essence, the balance that the ISP thinks is best for its economic interests (customer willingness to pay, versus the costs of provision).

These problems fundamentally arise, we believe, because the original Internet design principles failed to include socio-economic considerations, with the result that they aren't "designed for tussle" between the conflicting Internet stakeholders. Our aim is to translate the aspirational goal of "design for tussle" into more concrete Design Principles that are usable by protocol designers, network architects, and other members of the Internet community. In our analysis, we have found that, in the main, our new design principles are in fact adaptations and extensions of the original design principles:

*Original Design Principles * new socio-economic environment = tussle-aware Design Principles.*

The main contribution of this paper is the proposition of a new set of tussle-aware design principles. These are our initial proposals, which we are now seeking to validate and improve; hence we very much welcome comments on them.

This paper is structured as follows:

- Section 2 presents our primary contribution, proposing new design principles that we believe are better suited to the Internet's socio-economic tussles.

This is followed by two other brief contributions:

- Section 3 outlines how they relate to the existing well-known Internet design principles;

This research was supported by Trilogy (http://www.trilogy-project.org), a research project (ICT-216372) partially funded by the European Community under its Seventh Framework Programme. The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

978-1-4244-3437-4/09/$25.00 ©2009 IEEE

- Section 4 briefly sketches how the new design principles address some of the strains on the Internet mentioned earlier.

II. DESIGN PRINCIPLES

Here, we present a set of four new design principles for Internet evolution, developed with rapidly evolving socio-economic pressures in mind, which can be applied to the development of new architectures and technologies.

We make no claim that these design principles are complete – no architecture that meets all of them will, by itself, magically be the solution to all the Internet's ills. These principles are intended as guidance to designers for aspects to consider, in order to better enable the Internet architecture to adapt to rapidly changing socio-economic influences.

A. Information Exposure

The data (or transaction) that uses up scarce networking resources, through being sent (or acted on) should integrate control information (e.g. a metric) that reflects its resource usage in 'real time'. The control information should provide sufficient information about resource usage to support an efficient and accountable allocation of resources. The control information may be used as a variable in a pricing structure.

Some examples of the use of scarce networking resources are the forwarding of packets by a router ('data') and the creation or maintenance of a connection by a middlebox ('transaction'). The particular concern is those (economically significant) networking resources that are scarce, so their consumption by one user's data (or transaction) prevents another user consuming the resource; it is this externality (cost) that needs to be captured by control information associated with the operation. For example, where the scarce resource is a router's bandwidth, the control information is about the router's congestion. Other examples could be the shortage of table space for storing middlebox mappings, or processing power for handling router messages.

Additionally, the usage of resources which are not scarce will probably need to be accounted for, but this information does not need to be "integrated" (see below) with the data or transaction. Also, resources which are purely used and optimised within a single system are not relevant[1], e.g. resources purely at the end system (that are related to the data or transaction).

'Integrate' means that we believe this control information (e.g. a metric) must be within the data (transaction) and not some kind of slow management message, i.e. it automatically synchronises the information with the resource usage:

- In time (over what timescale)
- In space (over which participants)
- In function (for what activity)

[1] There are some marginal cases where it is unclear if it is a networking resource consumed by others or a purely local node optimisation, for example in a wireless mesh a burst of traffic that causes some device in power saving mode to have to wake up to deal with the burst.

'Reflects its resource usage in real time' emphasises the first of these points, i.e. the granularity of the information in time should reflect the time over which the data or transaction depletes the resources (i.e. the timescale it imposes load for). For example, for a data packet, the time is the period for the router's queue to drain (typically considered to be about 1 RTT). For a transaction, if the scarce resource is storage of state, then the timescale is the lifetime of the binding state.

For some types of resource, it is simply the existence of a data packet or transaction that consumes resource, regardless of its size. For others, the size of the data packet or transaction matters. For a data packet, the latter is typical (routers are 'byte congestible').

Practical considerations may spread the control information over time, e.g. ECN only allows one bit per packet and hence many packets must be monitored to get an accurate metric. One implication is that each node which uses up resource has to adjust the control information.

The second sentence in this Design Principle gives the reason for the control information ('support an efficient and accountable allocation of resources') and thus guides how detailed the information needs to be.

The third sentence reflects that market mechanisms are normally the right tool to mediate the allocation of resources amongst those competing to use them, and so are an effective way of achieving the goals of an effective and accountable allocation of resources. The actual pricing formula may be off-line and may be sophisticated.

There are some open questions with this design principle, which are likely to be resolved over time. For example, it may not be immediately obvious how much (processing) resource a transaction will consume, so what information is shared? Moreover, adding the integrated control information will itself consume some resource. We assume that the benefit is typically worth the cost, and such information will not be included where the cost outweighs the benefit.

B. Separation of Policy and Mechanism

Allow a network entity local choice according to its priorities (policy). Have a common protocol or method through which the policies interact to determine how networking resources are shared. Constrain conflicting policies via the Information Exposure Design Principle.

'Policies' are the implementation freedom that doesn't need to be standardised. This is both simple (the network entity knows what matters most to it, what its business models are, etc) and adaptable (adapt policy in light of its experiences). Implicitly, policies are independent of each other. On the other hand, the 'mechanism' is that which needs to be standardised.

A good example is Kelly's view of congestion control [3]. The mechanism is congestion pricing, which represent the externality cost that a packet has on other users. It is the end users' policy choice how they react to this information, e.g. they prioritise some flows (which continue unabated) over others (which adapt their traffic rate down), or they continue as before (and pay more, in some direct or indirect fashion).

Similarly, it is the policy decision of a network router when to congestion mark a packet (e.g. triggered by what level of traffic), as only it understands when its resources are starting to be depleted. Note that a congestion control approach where the network told the user what rate to run at (such as RCP [4]) would break this Design Principle.

Congestion marks also offer the opportunity for the network to create an admission control service, i.e. it uses the congestion marks to decide whether to admit or block a new flow admission request, in order to preserve the QoS of previously admitted flows. The network is free to set its own policies about how to translate the marks (the standardised mechanism) into its flow admission decisions [5]. Another example is DCCP [6]. It defines a three way handshake for connection set-up (the mechanism), which allows the ends to negotiate which congestion control algorithm (TCP, TCP friendly etc) to use for the data transfer phase (the policy). The choice is independent for each pair of endpoints. Yet another example is BGP routing. The "mechanism" is that an Autonomous System offers paths to IP prefixes to its neighbouring ASs. The "policy" is that each AS selects (from the advertised ASs) its "best" path towards a destination prefix, based on its own criteria.

"Constrain conflicting policies": However, policies need to be sensible – for example, the congestion control policy set by one user shouldn't ruin the service received by other users. We believe that sufficient constraint is provided by the accountability within the Information Exposure principle, i.e. the mechanism needs to integrate the control information (e.g. a metric) that reflects the usage of scarce resource by the mechanism's data (or transaction). For example, the policy decision about what granularity prefixes should be advertised/accepted (e.g. /16s or /32s?) will be constrained by the fact that too fine a granularity causes too high a cost in terms of amount of messages and processing. In other words, accountability forces the policies to be sane.

Another point is that one shouldn't expect that two different mechanisms will give the same answer. For example, we no longer try to synchronise the OSPF topology view with the BGP topology view in a single-layered routing architecture – we have two layers in the routing architecture, and have separate mechanisms in each.

C. The Fuzzy Ends Principle

Allow the endpoint to explicitly delegate some functions into the network, so the end is effectively a distributed system. This may imply some state in the network, which should be "soft and hinty".

The following list presents some examples of the sort of delegation that we are referring to. These are things that all could be done by the endpoint (client, server, or peer), but that the network could perform as a 'useful service' for the endpoint:

- Application prioritisation: e.g. a DPI box estimates traffic priorities, in order that the customer makes best use of the capacity under their contract.
- Protection: a firewall filters out certain types of traffic, e.g. adult content, file sharing, games, chat rooms, unauthorised access etc. This could be for consumers (parental control) or businesses (control employee usage of Internet). Similarly, firewalls restrict incoming access to authorised people only.
- Content caching and targeting: the network caches content from a server to speed up its delivery, or to help the content supplier optimise the content for a particular user.
- Finding an appropriate peer: a network oracle may be useful in a peer-to-peer system.

"Explicitly delegate" means that the endpoint has clearly delegated the function into the network, rather than the network imposing itself unannounced into the communications. Delegation is for a specific function or application. Note that "explicit" may or may not mean "voluntary". For example, a corporate network might insist that some functions are delegated from the endpoint to network nodes (middleboxes). Another example might be a broadband provider who insists some functions are delegated to it, as part of a service package. For trust and accountability reasons, it seems most likely that the delegation will be over a single administrative hop.

This principle seeks a compromise between network providers' interests to increase their profits by offering additional services to users, and the recognition that implementing functions in the network can affect the transparency of end to end communications. The Internet's stakeholders will tussle for economic and social control, but we want to try to stop an arms race between users and networks. We seek a more cooperative model where endpoints delegate, and the network helps. We also want to try to make sure that the system can evolve by drawing these functions back into the endpoints. Viewing the end point as a "distributed system" realizes these goals.

One open question for this Design Principle is whether it should place constraints on the kind of function that can be delegated. One view is that delegation should only be allowed to nodes acting as application-level intermediaries, which "are an integrated part of the application, correctly terminate the protocol stack and implement their functionality at layers above the Internet layer" [7]. An alternative view is that delegation doesn't just apply to application-level functions, for instance most middlebox functionality would fall into this category. The latter case leads to some loss of transparency of the network (it is no longer purely a packet delivery mechanism), and some loss of flexibility for the network and the endpoints (and their applications). To lessen this, we recommend that the network state is "soft and hinty" – in particular, that loss of the state should not stop the end system from being able to communicate.

Another open issue with this principle is how to extend it to permit more general delegation or relocation of control functions (a possible example might be BGP communities).

D. Resource Pooling

Allow sufficient pooling of resources to be effective. Ensure that the resource pooling mechanisms don't conflict with each other.

Resource pooling [8] makes the network's resources behave like a single pooled resource, i.e. separate resources appear to act as one large resource. So it implies the load (or a sufficient part of the load) can be "relocated" to a different resource or "spread" over several resources. "Relocation" and "spreading" can be in space or time, for example:

- In the analogue PSTN a phone call could set up a circuit using any of the empty channels on a switch;
- A packet switch makes available its entire capacity (not divided up into circuits), or at a later time, via buffering;
- Multihoming, multipath routing and traffic engineering pool together separate links or networks;
- Content Delivery Networks (CDNs) pool together the processing cycles, bandwidth and reliability of all their distributed servers;
- Network coding pools multiple messages into fewer messages over a pool of links;
- With swarming downloads (e.g. BitTorrent) each receiver pools multiple other peers receiving the same data to act as if they are one data source; and it pools all the network paths from these peers.

The benefits of resource pooling are:

- increased robustness against component failures;
- better ability to handle localized surges in traffic;
- maximized utilization.

"Allow sufficient pooling of resources to be effective" means there is enough resource pooling to bring about the benefits, i.e. there doesn't have to be unlimited relocatability. The "power of two choices" [9] suggests that in many cases it is enough if the choice is from a small set. Nor does it require that everything needs to be given a choice; sometimes it is enough if just a small fraction has choice. It may even be enough if the "spreading" doesn't explicitly involve a choice, e.g. with ECMP a specific flow follows a deterministic path, but overall it achieves reasonable load balancing.

"Conflict" (as in: "Ensure that the resource pooling mechanisms don't conflict with each other") exists if two resource pooling mechanisms work against each other, because they optimise for different criteria. For example, ISPs shift their traffic to minimise their peering costs, but peer-to-peer applications want to download from the peer reached over the best performing path. A "conflict" might also exist between two instances of the same resource pooling mechanism fighting to gain a bigger share of the pooled resource. We assume that conflicts can be handled via careful design and by applying the Information Exposure and Separation of Policy and Mechanism Design Principles.

III. RELATING THE NEW AND OLD DESIGN PRINCIPLES

This section briefly discusses where these new principles modify or extend the original principles, and identifies the socio-economic awareness that has been introduced in order to evolve these principles for the Future Internet. Reference [1] presents the following list of original Internet design principles and design choices:

- Self-Describing Datagram, the principle that control information should be carried within each datagram including any address / identifier information necessary to route datagrams between endpoints;
- Fate Sharing, the principle that state information should be stored within the entity where it is used, e.g. per-flow state should be kept at endpoints, with only state required for packet processing (the operation of the network) stored in the network;
- Layering, i.e. prescribing a hierarchical protocol structure with a clear identification and separation of the services provided by each layer to the layer above. In particular, the network layer is the unifying layer, supporting the heterogeneity of the higher and lower layers, i.e. 'IP over everything and everything over IP';
- End-to-End Argument, the principle restricting the implementation of functions in a network [10], [11].

Our Information Exposure Design Principle can be seen as extending the original Internet design principle of connectionless datagrams (signalling goes in-band, i.e. along with the packet) in two ways, in order to take account of the abstract goal of Design for Tussle. Firstly, by including the idea of accountability for usage of scarce resources, and secondly by recognising that transactions as well as packets consume resources.

Our Separation of Policy and Mechanism Design Principle can be seen as an evolution of the original Internet design principle of 'IP over everything and everything over IP'. That defined a simple IP layer to transfer data, with freedom for the applications above and the networking technologies below. Here we're concerned about a similar type of layering but for control – a common mechanism, with freedom over how the control information the mechanism provides is used ('above') and how the control information is set ('below').

Our Fuzzy Ends Principle is, unsurprisingly, closely related to the original Internet end-to-end design principle. The end-to-end principle in fact comes in two versions [7]: the "narrow" version of 1984 [10], which says a function should not be implemented in a lower layer (i.e. in the middle) if it cannot be completely and correctly implemented at that layer, except as a performance enhancement; and the "broad" version of 1998 [11], which says lower layers (the middle) should only provide very general services that can be used by all applications, as optimising support for one application hinders long-term system evolvability, application autonomy and reliability. The view that the Fuzzy Ends Principle should allow only delegation to nodes acting as application-level intermediaries is compliant with the "broad" and "narrow" version. The alternative view, which less restricts allowable delegation,

deviates from the "broad" version, but only in a confined, controlled manner so as to minimise loss of its benefits. It also extends the "narrow" version by providing two additional justifications for implementing functions in the middle: for business and social reasons, not just for performance enhancement as the original "narrow" version.

Finally, our Resource Pooling Principle can be seen as an extension of the original design principle of fate-sharing. Resources are pooled at the entities that have knowledge about them, for example state about the pooling of two end-to-end flows should only be held at the endpoints. The original desire for resilience in the Internet can be achieved through redundancy and diversity, rather than through super-reliable individual components.

IV. INITIAL VALIDATION OF THE DESIGN PRINCIPLES

In this section we attempt a very brief first sketch at validating the new design principles. We do this by returning to the two problems on the current Internet (concerning routing and resource control) discussed in the introduction and examine what our design principles have to say about appropriate solutions. Please note that this is on-going work. We welcome others who would like to try and validate (or disprove!) them, or have alternative new design principles.

To address problems in routing and resource control, we are developing a multi-path TCP protocol that instantiates the Resource Pooling principle. The sender and/or receiver are multihomed, and after an initial handshake, they exchange IP addresses, between different combinations of which sub-flows can be created. Packets that are lost from one sub-flow may be re-transmitted over other sub-flows. The congestion response of each sub-flow is coupled together (following the theoretical work of [12]). The expected benefits are not only increases in utilisation of the network, but also improved resilience through use of redundant paths. On-going implementations and theoretical work will validate the benefits more carefully.

We are also developing a framework for accountability for resource usage that instantiates the Information Exposure principle (in particular). "Congestion volume" is congestion integrated over time and data. It counts all the congestion a user suffers (or causes; they are equivalent) - however many flows are created, whatever route, whatever the activity factor, etc. The framework [13] contains the elements required to achieve accountability based on congestion volume:

- congestion information on each packet, i.e. ECN;
- a new mechanism such as re-feedback that gives the ability of any point in the network to calculate the congestion on the rest of the path;
- a policer at the first ingress, to catch those trying to cause more congestion than they're permitted to [14];
- a dropper at the last egress, to catch those trying to cheat by under-declaring the rest-of-path congestion;
- border gateways, counting the congestion volume in each direction between two adjacent ISPs. There would be inter-ISP contracts, similar to those for bandwidth;
- weighted congestion control. The end host runs an algorithm that reacts to congestion but weighted according to the priority of a particular flow. This achieves true end-to-end QoS.

V. CONCLUSION

In this paper, we have presented a new set of design principles for future Internet architecture, with particular focus on enabling socio-economic tussles between stakeholders, without going down the path of technical "hacks" that blight the Internet today. The four design principles have been shown to extend the original design principles of the Internet.

The validity of our proposed principles is now being explored through the development of new technologies and frameworks. This work is still in its early stages – and we fully expect our design principles to grow and evolve over time, as we explore new proposals and interactions, and as such we very much welcome feedback.

ACKNOWLEDGMENT

We thank all our colleagues on the Trilogy project, notably Robert Hancock and Ken Richardson, for their stimulating discussion leading to the development of the ideas presented in this paper.

REFERENCES

[1] D. Clark, "The design philosophy of the Darpa Internet protocols", in *Proc. ACM SIGCOMM*, Vancouver, BC, Canada, Sept. 1988.

[2] D. Clark, J. Wroclawski, K. Sollins and R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet", in *IEEE/ACM Transactions on Networking*, vol. 13, issue 3, pp. 462-475, June 2005.

[3] F. Kelly, "Models for a self-managed Internet", in *Philosophical Transactions of the Royal Society*, volume 358, 2335-2348, 2000

[4] N. Dukkipati, M. Kobayashi, R. Zhang-Shen and N. McKeown, "Processor sharing flows in the Internet", *Thirteenth International Workshop on Quality of Service (IWQoS)*, Passau, Germany, June 2005

[5] P. Eardley (ed.), "Pre-congestion notification architecture", IETF Internet-Draft (work in progress, August 2008)

[6] E. Kohler, M. Handley and S. Floyd, "Designing DCCP: Congestion control without reliability", in *Proc. ACM SIGCOMM*, Sept. 2006

[7] B. van Schewick, *Architecture and Innovation: The Role of the End-to-End Arguments in the Original Internet*, MIT Press, forthcoming 2009

[8] D. Wischik, M. Handley and M. Bagnulo Braun, "The resource pooling principle", in *ACM SIGCOMM Computer Communications Review*, volume 38, issue 5, pp. 47-52, October 2008

[9] M. Mitzenmacher, "The power of two choices in randomized load balancing", in *IEEE Transactions on Parallel and Distributed Systems*, volume 12, issue 10, pp. 1094-1104, October 2001

[10] J. H. Saltzer, D. P. Reed and D. D. Clark, "End-to-end arguments in system design", in *ACM Transactions on Computer Systems*, volume 2, issue 4, pp. 277-288, Nov 1984.

[11] D. Reed, J. Saltzer and D. Clark, Commentary on "Active Networking and End-to-End Arguments", in *IEEE Network*, vol. 12, issue 3, pp. 69-71, May/June 1998

[12] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control", in *ACM SIGCOMM Computer Communications Review*, volume 35, issue 2, pp. 5-12, April 2005

[13] B. Briscoe, A. Jacquet, T. Moncaster and A. Smith, "Re-ECN: Adding accountability for causing congestion to TCP/IP", IETF Internet-Draft Jan 2008

[14] A. Jacquet, B. Briscoe and T. Moncaster, "Policing freedom to use the Internet resource pool", to appear at ReArch workshop, ACM CoNEXT, Dec 2008