



---

ICT-216372

## **Trilogy**

### **Trilogy: Re-Architecting the Internet. An Hourglass Control Architecture for the Internet, Supporting Extremes of Commercial, Social and Technical Control**

Large Scale Integrating Project  
FP7 ICT Objective 1.1 – The Network of the Future

## **D7 Overall Architecture Including Design Principles**

Due date of deliverable: 30<sup>th</sup> June 2009

Actual submission date: 30<sup>th</sup> June 2009

Start date of project: 1 January 2008

Duration: 36 months

Lead contractor for this deliverable: BT

Version 1 date 30th June 2009

Confidentiality status: Public



---

Abstract

This document presents a proposal for the architecture of the Future Internet. The architecture has been developed by the Trilogy collaborative research project, whose focus is the development of the current Internet, and in particular its network and transport layers. The key challenges identified include scaling and flexibility in the global routing system, and resilience and resource management to support the needs and demands of emerging applications. As well as the strictly technical aspect, the new architecture aims to incorporate the necessary flexibility to adapt to changing economic and social stresses, the so-called “design for tussle”. While our architecture and design principles are presented here as an evolutionary step from the original Internet concepts, the implications for future network evolution are wide-ranging. Three architectural building blocks are proposed: multipath transport, multipath routing and an accountability framework.

Target audience

The primary target audience for this document is the networking research and development community, particularly those with an interest in the Future Internet. The material should be accessible to any reader with a background in packet switched network architectures. This document will also be of interest to those concerned with the interactions between network architectures and their economic, social and regulatory context, although specialist expertise in these areas is not a pre-requisite.

**Disclaimer**

This document contains material, which is the copyright of certain Trilogy consortium parties, and may not be reproduced or copied without permission. All Trilogy consortium parties have agreed to the full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the Trilogy consortium as a whole, nor a certain party of the Trilogy consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of its content.

**Impressum**

Full project title: Trilogy: Re-Architecting the Internet. An Hourglass control Architecture for the Internet, Supporting Extremes of Commercial, Social and Technical Control

Document title: D7 – Overall Architecture Including Design Principles

Editor: Philip Eardley, Toby Moncaster, BT

Project Co-ordinator: Alan Readhead, BT

Technical Manager: Philip Eardley, BT

This project is co-funded by the European Union through the ICT programme under FP7.

**Copyright notice**

© 2009 Participants in project Trilogy

## Executive Summary

The Internet as we know it today is orders of magnitude larger and more complex than was ever considered during its initial design several decades ago. In large part, it owes this success to the clarity and simplicity of the original engineering model: more sophisticated network architectures have come and gone, and in the meantime the Internet has continued to grow and evolve, often in unexpected ways. However, we may soon reach a stage where continued growth and innovation are stifled by the constraints of currently deployed networks; indeed, some commentators say this stage has already been passed. Trilogi is a three-year collaborative programme to research new networking technologies to address these problems, and this report describes the architectural framework that has been developed to guide the work, as well as architectural aspects of some detailed technical proposals.

This Internet paralysis has not been caused by any fundamental misconceptions in its original design. Rather, it is that the role of the network and the demands placed on it have expanded in scope so significantly: instead of serving a single community unified by a common purpose, the Internet now spans the global economy, where numerous stakeholders naturally seek to exploit it for their own purposes. Rather than attempt to redesign the Internet for a particular business model, we seek to develop an architecture that can adapt to the stresses of economic and social change, whichever direction they take, an approach christened by the original Internet architects as ‘design for tussle’. We also seek to extend the architecture to accommodate new concepts from the networking research community, to enable the Internet to meet new scaling challenges of network size and speed, and to support new classes of application. We believe this requires simplified control at all timescales

We approach the problem of developing a new architecture from two directions. On one hand, we have revisited the original Internet design principles, and re-evaluated them from the perspective of the architectural problems visible today. Our result is a set of four refined Design Principles:

- Exposure of Information
- Separation of Policy and Mechanism
- Fuzzy Ends
- Resource Pooling

These should be seen as complementing and extending the original Internet design principles, and are a first stage of capturing the abstract concept of ‘design for tussle’ as a more tangible engineering methodology. These four Design Principles were described last year in the Trilogi deliverable “Initial Overall Architecture”.

Our second avenue of investigation has been to take key novel networking techniques (especially applicable to the reachability and resource control problems), and analyse them for their architectural implications, especially points of compatibility and contradiction with the current Internet. We propose a baseline Trilogi architecture and have developed specific proposals (“building blocks”) in three areas within it. The baseline Trilogi architecture is comparable in scope to the current Internet network and transport layers, but with a subtly different internal structure. In particular, we divide the functions involving the networking infrastructure into two planes, for reachability and resource control, and distinguish them from the transport services to which the network appears as a black box. We also describe how it differs from the current Internet.

We have developed three architectural “building blocks”, which are:

- End-to-end transport protocols that can exploit multiple paths, for instance to take advantage of multihoming at endpoints in order to improve reliability and utilisation in the network.
- Multipath routing that allows routers to select multiple routes to a given destination, so as to improve reliability and resource pooling inside the network.



- 
- An accountability framework for resource usage that ensures end-users and network operators can be held to account for the impact their actions have on everyone else using the network.

While each can exist in isolation, a “loose coupling” of them significantly broadens their architectural footprint. We describe how they can complement each other and their potential interactions.

Overall we hope that our architecture can serve as a beacon for the evolution of existing networks. However, we do not expect that any clean-slate architecture will be deployed in the real world, so during detailed protocol design for each of the three “building blocks” we have carefully considered deployability. One factor here is to make small changes to existing protocols – but changes with significant impact for the party deploying them. Another issue we consider is the impact on the interface between the “application” and the network (the API).

Development of an architecture is essentially an artistic process: it cannot be formally derived from requirements or principles, or proven to be the unique solution of an engineering problem. Continuous validation of the design is therefore a critical process. We approach that question from two perspectives. Firstly, we consider how well our three “building blocks” match up to our four Design Principles and thus how well our architecture meets the goal of “design for tussle”. Secondly, we continue a detailed technical evaluation of our various protocol proposals, for instance through simulations, prototyping and discussion at the IETF.

## List of Authors

Toby Moncaster

Philip Eardley

Robert Hancock

Arnaud Jacquet

Pasi Sarolahti

Alan Ford

Rolf Winter

Francisco Valera

Pascal Mérindol

Iljitsch van Beijnum

Sébastien Barré

Andrea Bittau

Ken Richardson

Costin Raiciu

Alexandros Kostopoulos

Olivier Bonaventure

Marcelo Bagnulo

Damon Wischik




---

## Table of contents

<b>Executive Summary .....</b>	<b>3</b>
<b>List of Authors.....</b>	<b>5</b>
<b>Table of contents.....</b>	<b>6</b>
<b>List of figures .....</b>	<b>9</b>
<b>Abbreviations.....</b>	<b>10</b>
<b>1 Introduction .....</b>	<b>12</b>
<b>2 From Goals to Architecture.....</b>	<b>15</b>
2.1 Introduction .....	15
2.2 Socio-economic Goals.....	15
2.3 From Goals to Design Principles.....	16
2.4 From Design Principles to Architecture.....	18
2.4.1 Information exposure .....	18
2.4.2 Separation of policy and mechanism.....	18
2.4.3 Fuzzy ends.....	18
2.4.4 Resource pooling.....	19
2.5 Overall Trilogy Architecture .....	19
2.6 Differences between the Trilogy and Internet architectures.....	20
<b>3 Architecture Blocks.....</b>	<b>22</b>
3.1 Multipath TCP.....	22
3.1.1 Overview .....	22
3.1.2 Generic multipath architecture .....	22
3.1.3 Multipath TCP proposals.....	24
3.1.4 Two ended multipath TCP design .....	25
3.1.5 One-ended multipath TCP design .....	27
3.1.6 Multipath TCP security .....	27
3.2 Accountability Framework.....	29
3.2.1 Congestion notification .....	30
3.2.2 re-ECN.....	31
3.2.3 Policer.....	32
3.2.4 Dropper.....	32
3.2.5 Border gateway.....	33
3.2.6 Rate controller .....	33
3.2.7 Security of accountability framework .....	33

---

3.3	Multipath Routing .....	35
3.3.1	Intra-domain multipath routing protocols.....	35
3.3.2	Inter-domain multipath routing .....	36
3.3.3	Intra-domain and inter-domain path diversity combination .....	40
<b>4</b>	<b>Analysis of the architecture against the Design principles .....</b>	<b>42</b>
4.1	Multipath TCP.....	42
4.1.1	Resource pooling using multipath transport.....	42
4.1.2	Information exposure – making better use of available information.....	43
4.1.3	Separation of policy from mechanism.....	43
4.1.4	Fuzzy ends principle.....	43
4.2	Accountability Framework.....	43
4.2.1	Information exposure - congestion transparency using re-ECN.....	43
4.2.2	Separating policy and mechanism .....	44
4.2.3	Fuzzy Ends .....	44
4.2.4	Resource pooling.....	44
4.3	Multipath Routing .....	45
4.3.1	Resource pooling through multipath routing.....	45
4.3.2	Separation of policy and mechanism.....	45
4.3.3	Information exposure .....	45
4.3.4	Fuzzy ends.....	45
<b>5</b>	<b>Combining the architectural building blocks .....</b>	<b>46</b>
5.1	Overview .....	46
5.1.1	Routing .....	46
5.1.2	Transport .....	46
5.1.3	Accountability .....	46
5.2	A more detailed consideration.....	47
<b>6</b>	<b>API Perspective.....</b>	<b>51</b>
6.1	Congestion Signal APIs.....	51
6.2	Congestion accountability APIs .....	51
6.2.1	Minimal set-up .....	52
6.2.2	More advanced set-ups .....	52
6.3	Multipath TCP API .....	53
6.3.1	TCP semantics .....	54
6.3.2	Implementation of host policy.....	54
<b>7</b>	<b>Conclusions .....</b>	<b>56</b>
7.1	Future work .....	58



---

7.2	Concluding Remarks .....	58
<b>References</b>	.....	<b>59</b>
<b>Annex A</b>	<b>Design principles.....</b>	<b>62</b>
A.1	Information Exposure.....	62
A.2	Separation of Policy and Mechanism .....	63
A.3	The Fuzzy Ends Principle.....	64
A.4	Resource Pooling.....	65

---

## List of figures

Figure 1: The fat waist of the control hourglass.....	12
Figure 2: From goals to principles .....	18
Figure 3: Overall trilogy architecture.....	19
Figure 4: Multipath architecture: PM/MPS interface.....	23
Figure 5: Bijective path manager .....	24
Figure 6: Non-bijective path manager.....	24
Figure 7: The six elements of an accountability framework.....	30
Figure 8: Byte vs. packet congestible queues.....	31
Figure 9: Un-weighted congestion controller.....	33
Figure 10: Weighted congestion controller .....	33
Figure 11: A possible combination between intra- and inter-domain multipath routing .....	41
Figure 12: Network acting as a single resource pool (left) and two resource pools (right) .....	47
Figure 13: MPTCP exploits multiple addresses .....	48
Figure 14: MPTCP exploits path selection.....	48
Figure 15: Transparency of the whole path.....	50
Figure 16: Interfaces to the congestion signal.....	51
Figure 17: Simple congestion accountability API.....	52
Figure 18: More advanced congestion accountability API .....	53



## Abbreviations

3G	3rd Generation (mobile)
ACK	Acknowledgement (TCP)
API	Application Programming Interface
AQM	Active Queue Management
AS	Autonomous System
ASBR	Autonomous System Border Router
BGP	Border Gateway Protocol
CDMA	Code Division Multiple Access
CE	Congestion Experienced
DCCP	Datagram Congestion Control Protocol
DDoS	Distributed Denial of Service
DPI	Deep Packet Inspection
eBGP	external BGP
ECMP	Equal Cost Multipath
ECN	Explicit Congestion Notification
EIGRP	Enhanced Interior Gateway Routing Protocol
FIB	Forwarding Information database
FIN	Finish (TCP)
GRE	Generic Routing Encapsulation
iBGP	internal BGP
ID	Identifier
IDIPS	ISP-Driven Informed Path Selection
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IP	Internet Protocol
IPsec	Internet Protocol Security
IPTV	Internet Protocol Television
IRTF	Internet Research Task Force
IS-IS	Intermediate System to Intermediate System (protocol)
ISP	Internet Service Provider
LFI	Loop-Free Invariant
LLU	Local Loop Unbundling
LOCAL PREF	Local Preference (BGP)
LP-BGP	Longest Path BGP
MAC	Message Authentication Code (TCP)
MED	Multi-Exit Discriminator (BGP)
MpASS	Multipath BGP with AS Sets
MPLS	Multiprotocol Label Switching
MPS	Multi-Path Scheduler
MPTCP	Multi-Path TCP
NAT	Network Address Translator
OmTCP	One-ended multipath TCP
OS	Operating System
OSPF	Open Shortest Path First
P2P	Peer to Peer
PCN	Pre-Congestion Notification
PKI	Public Key Infrastructure
PM	Path Manager
QoS	Quality of Service

---

RCP	Rate Control Protocol
RED	Random Early Detection
Re-ECN	Re-feedback of Explicit Congestion Notification
RSA	Rivest, Shamir and Adleman
RST	Reset (TCP)
RTT	Round Trip Time
SACK	Selective ACK (TCP)
SCTP	Stream Control Transmission Protocol
SHIM6	Site Multihoming by IPv6 Intermediation
SSL	Secure Sockets Layer
SYN	Synchronise (TCP)
TCP	Transmission Control Protocol
TE	Traffic Engineering
UDP	User Datagram Protocol
VPN	Virtual Private Network
VoIP	Voice over IP
WiFi	Wireless LAN

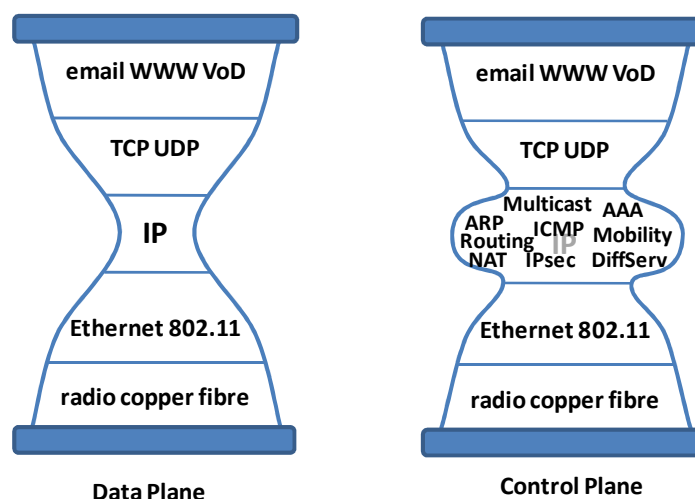
## 1 Introduction

The current Internet is a remarkable phenomenon, not only technically, but from an economic and social perspective as well. It has grown to become the network of choice for a huge variety of distributed applications, starting from email and file transfer, through a whole range of business-to-consumer and business-to-business e-commerce systems, all the way to online news, entertainment and social networking. The happy decision by the original Internet designers to implement a minimal, best-efforts data transfer capability has allowed the Internet to accommodate astounding innovation at the link and application levels.

Where the Internet has been less successful is in accommodating the conflicting economic interests of its participants, both between different operators of the network infrastructure and between the users and providers of network services. This problem has its roots in the restricted priorities of the designers [Clark88] rather than fundamental flaws in the design. It was defined in [Clark05] as the ‘tussle in cyberspace’, and arises increasingly often as the demands on the Internet increase.

One of the two overarching goals of the Trilogy project is therefore to develop an architecture for the future Internet that is ‘designed for tussle’: it can adapt in a scalable, dynamic, autonomous and robust manner to any local operational and business requirements. Our vision is that the architecture can thereby automatically adapt to the changes in society's demands on the Future Internet - it will not embed assumptions that unreasonably favour certain types of industry player, and indeed it will permit conflicting outcomes to coexist and evolve.

The Internet protocol stack is sometimes described as an hourglass with a “narrow waist”. This is actually the reason for many of the Internet’s elegant properties, such as simplicity, transparency and convergence, and therefore one of the key reasons for the Internet’s success. However, the narrow waist is really true for the data plane, whereas the story is different for the control plane, where the ‘waist of the hourglass’ is fat and getting fatter. The Internet has accumulated many control mechanisms in recent years, in piecemeal fashion, and with poorly understood interactions. The control mechanisms have often been “twisted” or “hacked”, especially to allow new business models such as for provider independence (using NATs), user control and differentiation (using deep packet inspection) and walled gardens (using firewalls).



**Figure 1: The fat waist of the control hourglass**

The second overarching goal of the Trilogy project is therefore to develop an architecture for the Future Internet that has a narrow ‘waist of the control hourglass’. In order to make progress we have confined our scope to reachability and resource control functionality at the network and transport

---

layers of the standard protocol stack. We develop new solutions that remove the known and emerging technical deficiencies, for example those outlined in the next two paragraphs.

In the routing area, as the scale increases and topologies become more strongly meshed (for example, because of both direct peering between edge providers and end-site multihoming), the lack of economic control over provider-provider interactions becomes more apparent. This lack shows itself technically as growth in router table sizes and churn rates, with consequent scaling worries about BGP, and the inability of individual operators to manage this load without harming end-to-end reachability.

In the resource sharing area, where the network capacity can be engineered to match reasonable expectations on long-term average data transfer requirements, the Internet copes well with the demands of best efforts applications. In recent years, these assumptions have been less and less valid. The Internet has become increasingly vital for more demanding applications (voice, physical infrastructure control, critical business operations), at the same time as peer-to-peer traffic has demonstrated its ability to absorb all available bandwidth and more. While integrated services approaches can solve these problems in specific scenarios, they cannot scale to general purpose Internet deployments because of the technical, and, even more importantly, administrative complexity of marshalling the resources of all the application users and network providers involved.

The Trilogy architecture is being developed to tackle problems such as those mentioned above. This deliverable summarises where we have got to with the development of the Trilogy architecture, and extends what was described previously [Burness09] [Trilogy08-D3]:

- We describe our high-level objectives for the Trilogy architecture. We describe how our high-level socio-economic goals have been transmogrified into practical and useful Design Principles, and thence how our architecture came out of these Design principles (Section 2)
- We hypothesise a baseline Trilogy architecture, which is comparable in scope to the current Internet network and transport layers, but with a subtly different internal structure. In particular, we divide the functions involving the networking infrastructure into two planes, for reachability and resource sharing, and distinguish them from the transport services to which the network appears as a black box (Section 2.5). Within this baseline architecture, we have worked on the following three components:-
  - We propose an end-to-end multipath transport protocol, which provides a mechanism for resource pooling. Resource pooling enables separate network resources to behave like a single large virtual resource, so achieving better resilience and efficiency. (Section 3.1)
  - We propose an accountability framework that is based on congestion volume. Accountability enables a rational basis for sharing resources amongst competing users, applications and businesses. (Section 3.2)
  - We propose a multipath capable routing system. It enables the exposure of path diversity to the multipath transport component, in order to achieve maximum resource pooling. (Section 3.3)
- To aid the development of our architecture we proposed four design principles in [Trilogy08-D3] – concrete guidelines to help a protocol designer or network designer achieve design for tussle. These are outlined in Section 2.3 and detailed in Annex A.
- We discuss our three architectural components in the context of the design principles – how design decisions were guided by them and how well they fit them. (Section 4)
- We discuss interactions between the three architectural components and describe how their “loose coupling” enables them to work harmoniously together. (Section 5)



- 
- We outline some API (Application Programming Interface) considerations for the architecture. (Section 6)

The Trilogy architecture is under continuing refinement and research. It will change as we validate it through our own activities and through feedback from other researchers and network designers – hence we welcome comments on the deliverable.

## 2 From Goals to Architecture

### 2.1 Introduction

The success of the Internet over the last 40 years – with few real changes deployed – can largely be attributed to the elegance of its major architectural concept: a simple, ubiquitous, transparent data delivery infrastructure, which can accommodate innovation at the link and application levels. We believe that this concept remains sound today and for the future. Hence the Trilogy architecture bears a striking similarity to the classic Internet architectural vision [Clark88] [Carpenter96], but with adjustments to try and incorporate our Trilogy Design Principles and thereby be cognisant of the wider socio-economic goals.

### 2.2 Socio-economic Goals

A key goal of the Trilogy project is to define an Internet control architecture that is adaptable to the socio-economic context of the current and Future Internet. In earlier deliverables [Trilogy08-D2] [Trilogy08-D3] we paid particular attention to the socio-economic ‘tussles’ that take place between the various stakeholders in today’s commercial Internet<sup>1</sup>, and identified a set of eight socio-economic goals from which we derived our Trilogy Design Principles. Here, we briefly restate those socio-economic goals with some minor amendments in the light of further experience, and then discuss how they link to our design principles.

Our set of eight socio-economic goals are:

- **Global Connectivity:** This goal is about providing the ability for end users to communicate with one another via the Internet, for both business and social purposes.
- **Business and Social Continuity:** This goal reflects the increased importance of the Internet as a business (and social) tool, with high demands on resilience and resource availability.
- **Application Innovation:** This goal is about encouraging innovation in the development of Internet applications.
- **Network Heterogeneity:** Here, the goal is to encourage flexibility and ubiquity of Internet access – independent of the underlying network technology.
- **Business Autonomy:** The goal here is to ensure that commercial enterprises enjoy sufficient autonomy and incentives to encourage them to invest in Internet network infrastructure, application innovation and service differentiation.
- **Utility Maximisation:** This goal aims to ensure the efficient use of scarce Internet resources in a manner that maximises the utility of end users.
- **Accessibility:** This goal is about facilitating simple access to the Internet and its services for all end users, from both a technical and market perspective.
- **Accountability:** This goal requires that an accurate and traceable record is available regarding the supply and allocation of Internet resources.

---

<sup>1</sup> The idea of ‘tussle’ between Internet stakeholders is a concept outlined in [Clark05]. The underlying premise is that the Internet, originally designed by, and for, a small not-for-profit academic user base has grown to become an all-pervasive commercial entity with many competing socio-economic interests – the ‘tussle’ – that requires the adoption of new or modified goals and principles to be applied to the process of Internet architecture design. Furthermore, the control architecture should be made as future-proof as possible so that other as yet undefined tussles can be played out between as yet undefined stakeholders.



Some of these are fundamentally unchanged from the original goals of the Internet whilst others emerge as a direct consequence of its commercialisation, especially Business Autonomy, Utility Maximisation and Accountability. Unsurprisingly, these goals may be in conflict, for example Business Autonomy reflects the interests of commercial enterprises whilst Utility Maximisation reflects the interests of end users. This highlights the tension between maximising the social (or collective) benefit of the Internet whilst providing opportunities and incentives for commercial organisations to return a profit. The ‘tussle’ is a playing out of these conflicting interests and Trilogy aims to provide an architectural framework within which they can be fought without resort to a distortion of the Internet’s control architecture.

In the early days of the Internet, the user community was such that administrative control or peer pressure were sufficient to constrain stakeholder behaviour in order to maximise the general good. Commercialisation of the Internet has meant these mechanisms have been replaced by market forces as the central means by which conflicts between different interests and stakeholders are resolved.

### 2.3 From Goals to Design Principles

[Clark88] shows how the original technical goals of the Internet were reflected in a number of design principles;

- 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;<sup>2</sup>
- Layering, the principle that 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 that “certain required end-to-end functions should only be performed by the endpoints (although supporting low level mechanisms are justified if they provide performance enhancements)” [Saltzer84]<sup>3</sup>.

In [Trilogy08-D3] we extended this approach to derive a number of new design principles from our new set of socio-economic goals. Here, we provide some more substance and justification to this derivation.

The first point is that we recognise that some goals (and the tussles they encompass) are best resolved outside the control architecture - via market competition supported by an appropriate competition policy and/or regulation. In these areas, particularly the Application Innovation, Network Heterogeneity and Accessibility goals, the aim is to adopt design principles that ensure a neutral architecture rather than unduly biasing it in favour of one stakeholder group over another. In other words, the Internet should be equally accessible to all end users and content providers, so that end users are capable of accessing content without limitation or restrictions imposed by network providers.

At the time of writing [Trilogy08-D3] we were unclear about whether there to list a fifth original principle, Distributed Control, stated as follows;

---

<sup>2</sup> This avoids the need to provide mechanisms to discover and reinitiate remote state information in the event of a failure.

<sup>3</sup> This principle only received its definitive formulation in [Saltzer84] but was clearly an original technical design principle, a logical extension of the fate sharing and layering principles.

- Distributed Control, the principle that autonomous entities within the Internet should be responsible for management of their local policy and data.

Its origins lay in the **technical** separation of the Internet into Autonomous Systems. The commercialisation of the Internet has added the **commercial** dimension to this separation, which today manifests itself as a tussle over routing policy through the use (and arguably abuse) of BGP. Consequently, we see policy and mechanism mixed up together. Going forward, we believe these should be separated from each other. This is one example that led us to our first Trilogy Design Principle, ‘Separation of Policy and Mechanism’ – as a restatement and generalisation of the Distributed Control principle. Furthermore, there are times when stakeholders are prepared to delegate control to other stakeholders, which leads to our ‘Fuzzy Ends’ principle, a refinement of the original End-to-End Argument.

Maximising the utility of end users through the most efficient use of network resources, whilst also achieving high resilience are key goals. Both can benefit from the pooling of resources, hence our ‘Resource Pooling’ principle.

Finally, as alluded to above, a consequence of the commercialisation of the Internet is that tussles between stakeholders tend to be resolved by market forces. In particular, the tussle over scarce network resources is a tussle that needs to be resolved through the alignment of market-based incentives for both network providers and end users. Our ‘Information Exposure’ principle provides the technical underpinning of the use of market-based price mechanisms to achieve this alignment whilst recognising that the mechanism itself may reside outside of the actual control architecture.

This leads us to our four Trilogy Design Principles:

- *Separation of Policy and Mechanism Principle*, the principle that higher level policy decisions – the implementation freedom – are separate from the standardised mechanisms for implementing control through the exchange of information. This most obviously links to the Business Autonomy goal.
- *Fuzzy Ends Principle*, the principle that endpoints are able to explicitly delegate some functions to the network. This also links to the Business Autonomy goal, and more indirectly, the Accessibility goal.
- *Resource Pooling Principle*, the principle that resources in the network should be able to be pooled, so that they can effectively be utilised as a single resource, in order to improve the effectiveness and efficiency of the network. This links to the Utility Maximisation goal, and also the Business and Social Continuity goal.
- *Information Exposure Principle*, the principle that sufficient information about resource usage should be exposed to support an effective and efficient allocation, in a timely fashion. To achieve this, the information is integrated into the data (or transaction) that uses up the scarce networking resource. This links to the Accountability goal and, indirectly, to the Utility Maximisation goal.

The Design Principles are explained in greater detail in Annex B.

Figure 2 shows the linkages between the socio-economic goals and the Design Principles. It should be noted that the linkages tend to be to the goals that have a more socio-economic focus, of Business Autonomy, Utility Maximisation, Accessibility and Accountability, whilst the other goals are covered by the original design principles.

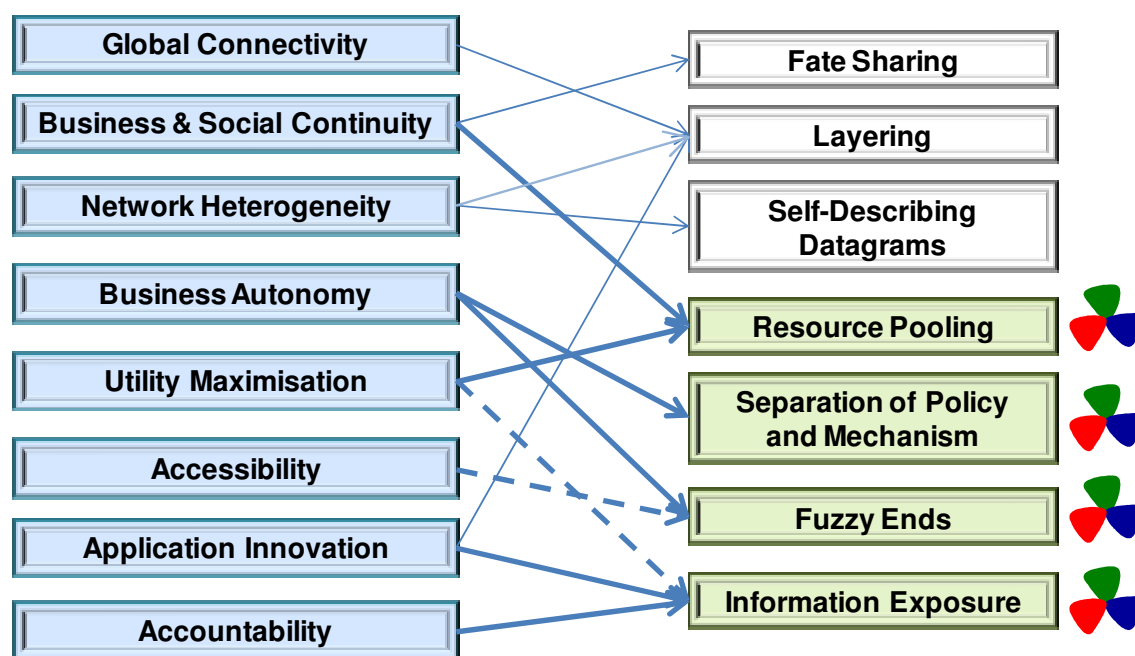


Figure 2: From goals to principles

## 2.4 From Design Principles to Architecture

### 2.4.1 Information exposure

We have mainly considered resources used when forwarding a packet. Congestion occurs when the “demand” for these resources exceeds the “supply”. For most types of link this is when the amount of bytes in packets exceeds the link’s bandwidth. Currently, visibility of congestion information is limited in two main ways: only the ends points can see it easily; and it is easy to cheat by mis-declaring congestion. To try and solve these issues “hacks” have been introduced such as DPI and TE management, but they are somewhat complex and limited. Instead we want visibility (transparency) of information about resource congestion. We are pursuing these ideas at the IRTF and IETF [Briscoe09b, Briscoe09c].

### 2.4.2 Separation of policy and mechanism

This design principle is particularly about modularity of control – a common mechanism with freedom over how the control information is used and set. To continue the congestion example, it means that:

- Rate control by sources is governed only by local policies and network-derived load information signalled along the delivery path. The utility functions are private to each user, and so do not need to be coordinated between themselves and the network;
- Each forwarder (link) decides when it will indicate congestion, based on its own private calculation, for example whether it uses RED [Braden98] or perhaps a virtual queue [Eardley09].

### 2.4.3 Fuzzy ends

Delegation of functions from end-points into the network is not something we’ve thoughts about in detail yet, although we hope to study proxy versions of Multipath TCP, whereby better informed nodes in the network can make the multipath decisions. Also end-nodes could tag flows to allow

routers to make more intelligent multipath routing decisions on their behalf. Additionally, it seems feasible that congestion accountability could be applied using proxy boxes instead of the end hosts.

#### 2.4.4 Resource pooling

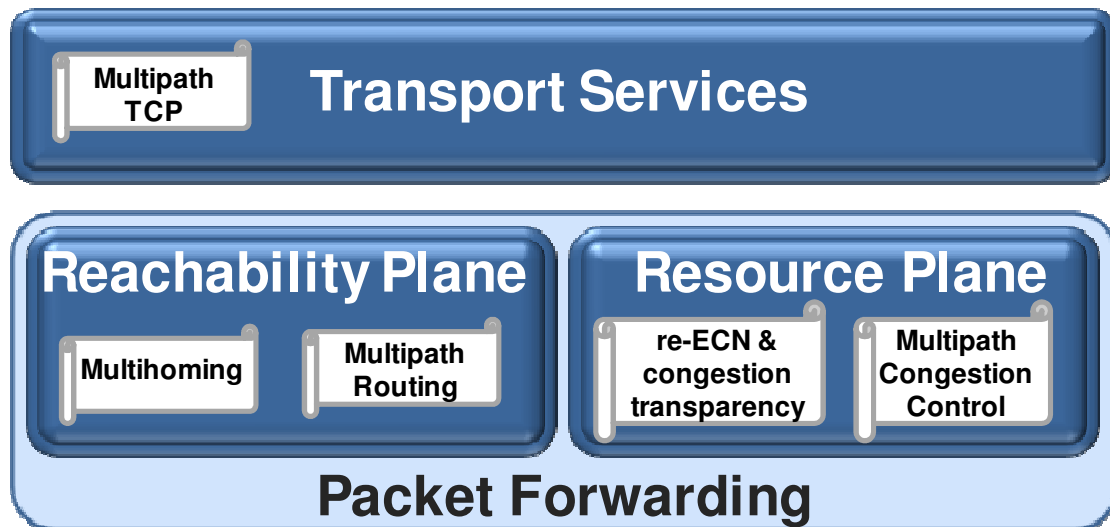
We are looking at two techniques for resource pooling. Firstly, we propose Multipath TCP implementations that use multiple paths to support a single end-to-end connection. This gives benefits in terms of improved resilience and capacity, both for individual flows and for the overall admissible set of flows as a whole.

Secondly, multipath routing, where the underlying routing system can maintain multiple routes for a single destination. If the routes are reasonably disjoint, then again this gives resource pooling benefits.

### 2.5 Overall Trilogy Architecture

The Overall Trilogy architecture is divided into three parts:

- The reachability plane: responsible for hop-by-hop outgoing link selection and hence enabling network-wide reachability.
- The resource plane: responsible for deciding how the transmission resource on each link is apportioned between packets.
  - Together, the reachability and resource planes form the packet delivery service.
- Transport services: functions, such as reliability, flow control or message framing, which are totally invisible to the packet delivery service.



**Figure 3: Overall trilogy architecture**

The reachability plane is responsible for hop-by-hop outgoing link selection. The destination address (in its locator role) is the key piece of information needed to route each packet (although routing may also be explicitly influenced by path selector bits and service class identifiers, i.e. other identifiers visible at the packet level). Every allocated locator is reachable by default and ideally is drawn from a single global namespace (this is the only approach totally compatible with the packets being fully self-describing, although NATs permit unmanaged or private addresses). Functions such as (D)DoS protection must be implemented at the receiver end system. We assume that the reachability service is implemented by a set of autonomous, interconnected administrations or domains. The internal



---

structure of each domain may be non-trivial, but is externally invisible: all that is exposed is information about which locator spaces are reachable and under what circumstances.

The resource plane decides how transmission resources on each link are apportioned between packets, and is also responsible for controlling the rate at which packets enter the network from applications. Information about resource allocation along the path can either be implicit (delay or loss) or explicit ('marking' in the packet header); end systems either measure the implicit path properties or read the explicit information, and in response modify their sending rates over the different paths exposed by the reachability plane. The allowed set of responses is left very open at this stage, and specific issues of accountability are discussed further in Section 3.2. The fundamental packet delivery service is best-effort: the network delivers packets to their destination, without guarantees on ordering or throughput. However, there is an implicit requirement that an end system generating a sequence of packets can rely on path consistency, at least over a scale of a round trip time or more.

Transport services are the means by which the packet delivery functions are actually exercised. They are implemented in a pure end-to-end fashion and define the communications service offered to applications (email, messaging, file sharing, multimedia, ...). Included are functions such as reliability, flow control or message framing, which are totally invisible to the packet delivery service.

## 2.6 Differences between the Trilogy and Internet architectures

First it is worth pointing out a critical way in which they are the same: the same four implicit layers of: networking technology, IP packet delivery, transport, applications. This brings all the well-known benefits such as application freedom and innovation, easy evolution of networking technologies, fate sharing, connectionless datagrams, etc.

However, there are several subtle differences.

We have separated the reachability and resource planes, which are not divided on the current Internet architecture. One reason is that the relevant identifier spaces are different. The key identifier space is the destination locator, which is handled in detail only by the reachability plane, whereas the resource plane treats locators as opaque tags which only have to be tested for equality, in order to test for path consistency. (The transport services use endpoint identifiers that label the communicating parties, but these are totally independent of the locators used in the reachability plane, and indeed need not be global or coordinated.)

Also, we recognise that either the reachability or resource planes may be shrivelled or vestigial. For instance, a future all optical network with wavelength switching may have no queues or AQM (consisting of a reachability plane only) or the network may be layer 2 rich (e.g. gigabit-Ethernet) with almost no IP-aware nodes (consisting predominantly of a resource sharing plane). Separation also enables networks to develop and evolve in each area, without being hampered by unexpected feature interactions.

A second difference is that rate control is part of the resource plane, whereas in the current Internet architecture it is a function of transport protocols. Rate control is about the use of the resource plane, in effect the source is adapting to reflect the resource availability over the entire path – fundamentally the same as each router dropping or marking packets according to the resources available on each link. As another example, the resource plane includes policing or admission control by a domain's ingress border router.

The interaction between the reachability and resource plane is another fundamental difference between the Trilogy architecture and the current Internet architecture. The reachability plane exposes multiple paths to the resource plane and it is up to the resource plane to determine how to distribute the load among the different paths. So, through this new interface between the two planes, the Trilogy architecture provides a richer reaction to congestion. In the current Internet architecture, the only (safe) reaction to congestion is to reduce the sending rate, whereas in the Trilogy architecture, the sender can move traffic between paths, reacting to congestion by diverting load onto less congested

---

paths. The transport services plane now has to deal with the performance impact of packets arriving over multiple paths (e.g. lack of ordering preservation).

Another difference is that we have added “congestion accountability” to the resource plane. This relies on “congestion transparency”: the ability of any node in the network to see the congestion caused by the packets it forwards. Such transparency enables, for example: sources to adapt their rates and perhaps shift transmission onto other networks, according to the network conditions as weighted by the data’s utility; and networks to adapt, for example by traffic engineering. An accountability framework ensures that parties are honest, i.e. don’t obfuscate the congestion.

Finally, we have boosted the reachability plane with multipath routing beyond the current limited capability of ECMP, for instance.

Having outlined the overall Trilogium architecture, we now look in more detail at the three specific architectural refinements that we’re developing: multipath TCP, the accountability framework and multipath routing. We initially present them as independent building blocks, whilst Section 5 looks at the implications of combining them.



## 3 Architecture Blocks

As mentioned above, we have been working on three major architectural blocks within Trilogy: Multipath TCP, an accountability framework and Multipath Routing. These are explained in some detail below and are also discussed in [Trilogy08-D4] [Trilogy08-D5].

### 3.1 Multipath TCP

#### 3.1.1 Overview

The resource pooling principle [Handley08] sets out a goal that it is a natural evolution of the Internet for “a collection of resources to behave as a single pooled resource”, where resource could be bandwidth, processing, etc. Resource pooling increases robustness against failures, provides better ability to handle localised surges, and improves resource utilisation. Combined, all these factors enhance user experience and efficiency of resource usage. The pooling of bandwidth is the most obvious exploitation path, and this is the aim of Multipath TCP (MPTCP). MPTCP supports the use of multiple paths between source and destination, thus bringing the main benefits of increased robustness (as the ensemble of paths has greater reliability than any individual path), higher throughput and better utilisation of the network.

Although the current Internet’s routing system only exposes a single path between a source-address pair, more and more hosts, especially mobile hosts, have multiple addresses that could be used to generate multiple paths to other hosts on the Internet. Thus, multipath transport could be deployed in the current Internet without requiring an upgrade to the routing system. Section 3.3 considers modifications that expose multiple paths.

In order to implement resource pooling for bandwidth, in theory there is a choice about what layer to implement it. Below the transport layer, shim6 for instance could be modified to automatically send packets over multiple paths; ECMP does this too, but reroutes the whole flow. Implementing above the transport layer – at the application level – can be effective, and is already used by applications such as BitTorrent. The problems here stem from the fact that congestion control is difficult to get right, and re-implementing it for every application is not practical; also, to universally reap the benefits of multipath means modifying each application, which is not practical.

Although layers higher or lower than the transport layer may implement sending rate adaptability to network conditions (such as streaming protocols that switch to a different video quality depending on available bandwidth), in general this capability is only available in transport protocols such as TCP, SCTP and DCCP. When implemented in applications, the capability is often effective, but not very efficient. For instance, BitTorrent will effectively use multiple paths, but the protocol incurs significant overhead accomplishing this and may end up finishing a transfer inelegantly as it waits for the final data blocks to arrive from slow peers, or re-request these blocks from faster peers.

It seems natural, therefore, to implement multipath functionality directly at the transport layer. Additionally, to have immediate impact, the multipath protocol must be designed with care such that it is compatible with the current Internet, and thus easily deployable. As TCP is currently the most widely used transport protocol, we have chosen to adapt TCP to become multipath capable and are now pursuing this at the IETF [MPTCP09].

#### 3.1.2 Generic multipath architecture

The basic idea of the architecture is to decouple the mechanisms of Path Management (PM) from the decision of what path to use for a particular packet (MPS, Multi-Path Scheduler). The main motivation for this decoupling is that there is more than one possible technique for each.

The **Path Manager** responsibilities are: to discover the available paths between a given pair of hosts; to define a Path Index for each path; to publish the Path Indexes to the MPS (perhaps with information

about known path properties, which may influence the MPS); to handle necessary changes in packet headers resulting from the use of multiple paths (e.g. in 2-ended multipath TCP, it translates from the address exposed to the application to the address for the specific path). Potential techniques to expose multiple paths to the end-host include: multiple address/port pairs, Shim6, next-hop selection (for local outgoing path selection), tunnelling and even configuration in advance - other examples are provided in [Trilogy08-D4].

The **Multi-Path Scheduler** is responsible for deciding on which path to send each packet, normally based on congestion information from each path, i.e. congestion control over the multiple paths. The MPS listens to events received from the PM, and sets its outgoing queues accordingly. The MPS can use fewer paths than announced by the PM, but naturally it cannot use more paths. As well as the two-ended and one-ended Multipath TCP variants, it may be possible to extend to a UDP or DCCP-based multi-path scheduler.

Figure 4 shows a high level view of the multipath architecture, with its two functions: PM and MPS. The main abstraction provided by the Path Manager is the *Path Index*. A Path Index is associated with a path, independently of the method used to send data on the path, which remains hidden by the Path Manager.

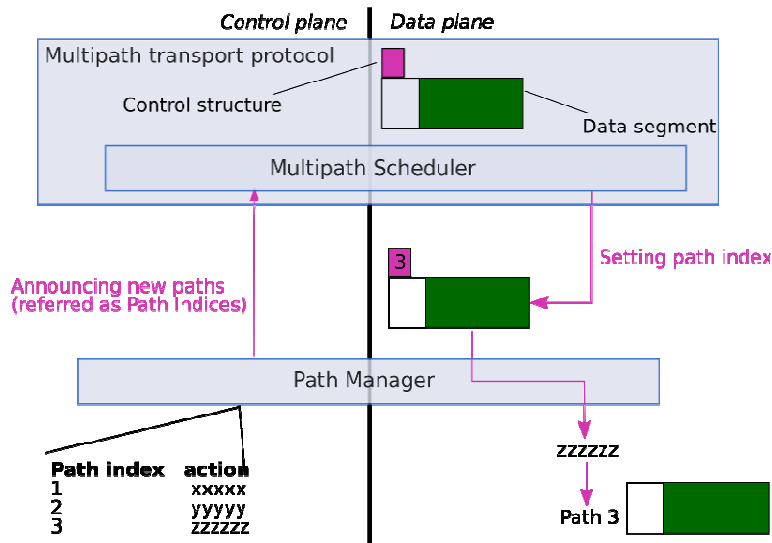


Figure 4: Multipath architecture: PM/MPS interface

**Outgoing packets**

In Figure 4, the path of an outgoing packet through the two building blocks of the architecture is shown. Here the MPS has learnt that 3 paths are available, and can be selected through path indices 1, 2 or 3. For the current packet, path 3 is selected. The MPS attaches that value to the control structure of the packet (not in the packet itself since the information is not supposed to go to the network), Finally the packet arrives in the Path Manager, which uses its mapping table to understand that action “zzzzz” (e.g. address translation) must be applied for that packet, before it is sent onto the wire.

**Incoming packets**

This architecture described so far allows for sending packets through different paths, but if host A communicates with host B, the outgoing paths of host A are the incoming paths of host B. Based on this, path managers can be separated in two categories:

- **Bijjective Path Managers:** These Path Managers (example in Figure 5) maintain an association (or “bijection”) between the incoming path and the outgoing paths, such that the MPS perceives each pair of paths (incoming and outgoing) as a single bidirectional path, and so the MPS can understand that an Acknowledgement received on one path acknowledges a segment sent over the corresponding outgoing path. The requirement is that the PM can tag incoming packets with the same Path Index that outgoing packets get on the same path. Examples of Bijjective Path Managers are the one included in two-ended MPTCP proposal, and the Shim6-based path manager.
- **Non-bijjective Path Managers:** These are path managers (see example in Figure 6) without an unambiguous mapping between incoming and outgoing paths. Hence an MPS that needs congestion feedback on its outgoing path must provide its own mechanisms to disambiguate information on the incoming paths. On the other hand, the PM perhaps may provide greater flexibility with the control of paths and no path negotiation is required with the peer. An example of non-bijjective Path Manager is the one used by one-ended MPTCP (see Section 3.1.5).

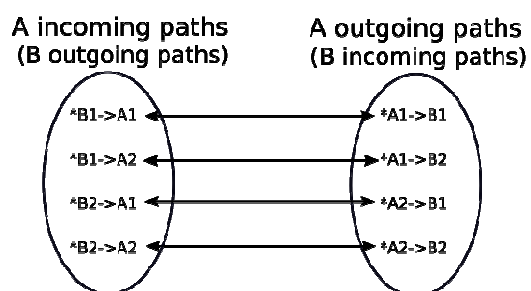


Figure 5: Bijjective path manager

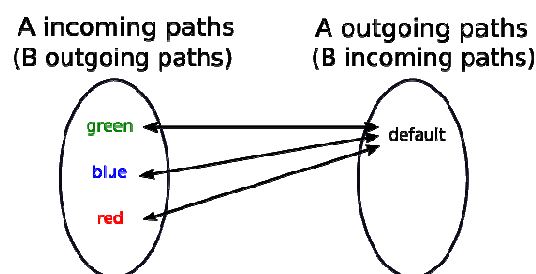


Figure 6: Non-bijjective path manager

### 3.1.3 Multipath TCP proposals

We have developed multipath TCP – adapting TCP so that it can use several paths. To ease the deployment of multipath TCP and so speed the realisation of benefits from resource pooling, existing applications should work unmodified. This means that the existing interfaces of the socket API must not be changed, and that standard calls to this API should result in the instantiation of multipath connections. This does not preclude a more advanced API from being deployed, however, and utilised by multipath-aware applications, to support finer-grained control of path selection according to application needs and preferences, discussed in detail in Section 6.3.

Research is ongoing on the most appropriate way to implement congestion control for multipath flows. One straightforward technique is simply to run the standard TCP congestion control algorithm separately for each subflow, and multiplex data from the different paths after applying the segmentation function already present in the transport protocol. This adapts to the aggregate available capacity. However, we believe that it is better to couple the congestion control loops of the subflows together – so that the traffic rate on a congested path is decreased more aggressively whilst increasing faster on a less congested path. A further general point is that we wish the multipath protocol to support multiple congestion control algorithms (as a separation of policy and mechanism). All the protocol needs to do is to inform the congestion control algorithm of the congestion status of each path (e.g. which of the paths dropped a packet or received ECN marks, and which path delivered a packet)

We envision two deployment paths for multipath TCP. One is to implement the new protocol in networking stacks of major operating systems (such as Windows, Linux, etc). In this case the

---

connection would work only if both ends are multipath TCP capable, and at least one endpoint is multi-addressed. Section 3.1.4 describes this work.

Another deployment scenario is at major web hosting servers, Content Distribution Networks or other providers, such as Google or Amazon. These would implement a variant of multipath TCP that requires only the sender to change. This applies to hosts that are multihomed but typically single-addressed. The assumption is, in this case, that the web providers would be faster at implementing the new protocol than large OS vendors, and many are already multihomed in a suitable way. Section 3.1.5 describes this work.

These two deployment scenarios constrain the design in different ways, and the results are significantly different. In the following sections, we provide a brief overview of the resulting protocol designs while spelling out the reasons for our choices. Two protocol designs are presented: firstly the two-ended MPTCP, where both endpoints are aware of the Multipath TCP and so can negotiate any combination of available paths. Secondly, the one-ended design is presented, which is designed for use by large sites that can be multihomed but single-addressed, allowing rapid realisation of resource pooling benefits with only one end requiring upgrading.

In terms of the generic multipath architecture above, the issues described in the following sections are nearly all part of the PM – MPS issues are the subject of on-going work which will be described in later Deliverables.

### 3.1.4 Two ended multipath TCP design

The “two-ended” multipath design [Ford09a] provides multipath TCP capability when both endpoints understand the necessary extensions to support MPTCP. This allows endpoints to negotiate additional features between themselves, and initiate new connections between pairs of addresses on multihomed endpoints.

#### *Basic operation*

The goal of a multipath TCP flow is to take a byte-oriented stream of data from the application and splits it across multiple “subflows”, i.e. different TCP (sub)connections that make up a larger multipath TCP connection. To an application, a MPTCP session looks no different to a standard TCP connection – the addition and combination of new paths is handled at the transport layer. It is expected that an extended API (see Section 6.3) could be applied to give MPTCP-aware applications more flexibility.

A particularly notable issue is that regarding sequence numbers: TCP has a single sequence number for the data, and assigns some parts of it to the signalling mechanisms to give them reliability (SYN and FIN). Taking these sequence numbers and sending them on different paths is the simplest thing to do, but has several issues:

- Each of the TCP subflows will look like a fragmented TCP connection over the wire, and may be blocked by middleboxes that reconstruct the TCP stream, such as traffic normalisers.
- When a packet is lost on one path, the cumulative ack in TCP will be blocked, hiding the receipt of packets on other paths. This could be resolved by using SACK, but there remains a further issue, in the next point.
- When a lost packet is retransmitted on a different path, the sender cannot know for sure which path really delivered the packet, and so cannot make a fully-informed decision about scheduling future packets.

So when designing MPTCP we decided that each subflow has its own sequence number space, and there is an additional data-level sequence number, presented in a TCP option. The use of TCP options seems the most appropriate way of including metadata about packets without interfering with the payloads although it does carry a penalty in terms of increased packet overhead. Each TCP segment now needs to identify both the subflow and the data (flow) sequence number. The receiver uses the



---

data sequence number to reorder the application data. The receiver only acknowledges subflow sequence numbers. The sender can infer from this exactly which data segments have arrived, and on which path, and can identify which data needs to be retransmitted upon packet loss. The inherent reliability of TCP exists at the subflow level.

For any given subflow sequence number, a single data sequence number can be mapped on it (the mapping is an injective function). This is to ensure compatibility with traffic normalisers, which may replay cached data if they see the same (subflow) sequence number, whilst ensuring the sender knows which data has been received.

This also means that, if a data segment is lost on subflow A and retransmitted on subflow B, subflow A must keep retransmitting the segment until it is acknowledged, in the normal way, even though the data segment may have arrived already at the destination (as it cannot map new data on the same sequence number). We considered using an optimisation to instruct the receiver to skip such holes and thus move forward to the next expected sequence number – which we termed the resync packet – but this would naturally create holes in the subflow sequence number space which could also confuse middleboxes.

### *Connection management*

A multipath connection starts as a normal TCP connection with an additional option that indicates that the sending host is multipath capable, and it includes a token that the sender uses to locally uniquely identify this multipath connection (which can be seen as being analogous with a port number). If one of the endpoints does not support multipath, or if middleboxes drop new options, the connection falls back to a standard TCP connection.

Once the connection has been opened, and thus identifying tokens for each endpoint are exchanged, either endpoint can create additional subflows by initiating the three-way handshake on a different path (see below for path discovery mechanisms). These SYNs will contain the token the endpoint received upon connection initiation, which is included to allow correct association of the subflow to the overall Multipath TCP connection. The source and destination ports of these subflows no longer have the same meaning as for regular single path TCP, as the token is also used for demultiplexing (it is likely that they will use the same destination port as the initial connection to ease NAT and firewall traversal, and to assist with traffic engineering).

In terms of **Error! Reference source not found.**, this means that a single initial path is used (as if no multipath is present at all) and subsequently the Path Manager can notify the MPS of the set of path indices corresponding to other paths to this destination (perhaps after a phase where the PM discovers the other paths).

Subflows can be added or removed during the lifetime of the connection, perhaps to accommodate new available paths. Removal is normally done using the FIN/FIN ACK exchange. However, to ensure graceful connection shutdown even when some paths have failed, we need an additional DATA FIN which acts as a connection level FIN, indicating that the sender has no more data to send.

### *Path discovery*

Two-ended multipath TCP discovers paths through the presence of multiple routable IP addresses on a host. A host is aware of its IP addresses, and can attempt to initiate new connections to the other endpoint using these new addresses. An endpoint can also inform its peer of alternative addresses it is reachable on, in the event that it cannot itself initiate connections due to the peer being behind a NAT or firewall.

Multipath TCP should be able to use any type of path diversity. Proposals like shim6 or modifications to the routing layer can provide such paths.

---

### 3.1.5 One-ended multipath TCP design

The One-Ended Multipath TCP (OmTCP) [Beijnum09b] design takes advantage of the fact that all the decision-making relevant to congestion control is done by the sender. The receiver simply sends acknowledgments back to the sender after receiving data packets. With OmTCP, the sender distributes regular TCP packets over multiple paths, recording which packets (parts of the sequence number space) were sent over each path. Then, when the receiver acknowledges the receipt of data, congestion control can be executed for the specific path that delivered the data.

Using multiple paths leads to high levels of packet reordering, so acknowledgments of new data may be delayed significantly and “duplicate ACKs” could trigger the fast retransmit algorithm. OmTCP avoids such issues through the use of selective acknowledgments (SACK), which is widely implemented throughout the Internet. The acknowledgment of out of order data with SACK makes it possible for congestion control to progress for each path individually without delay. Also, with SACK, the fast retransmit algorithm can be implemented per path, so reordering between paths no longer triggers fast retransmissions and the accompanying congestion window reduction.

OmTCP requires a mechanism to allow an OmTCP sender to direct TCP packets over different paths. The initial proposal is that OmTCP either sends packets through different outgoing interfaces (if the interfaces allow for the use of the same source address) or sends packets to different next hop routers. Later, more advanced mechanisms may be developed, such as integrating OmTCP with shim6 or the use of a path selector value in packets that is subsequently taken into account by routers when making forwarding decisions (as in ECMP).

### 3.1.6 Multipath TCP security

Multipath TCP allows multiple flows to participate on one logical connection, which raises the security concern that an attacker might be able to join a connection and compromise its integrity. To control who can join a connection, multipath TCP requires presenting a 64bit token on joining (as described in Section 3.1.4) - a random number generated during the first connection handshake, which cannot easily be guessed by attackers. Attackers that can eavesdrop on the network can however learn this token and can still join and compromise the connection. We developed “tcpcrypt”, a more robust solution that can also stop passive attackers with eavesdropping capabilities from joining multipath connections. Tpcrypt is an extension to TCP that can be applied independently of multipath TCP. We first describe the essentials of tpcrypt, and then how it works in a multipath TCP context. Tpcrypt is described here for the first time; work is on-going and progress will be described further in later deliverables and publications.

Each tpcrypt host generates an ephemeral RSA key pair to establish a session key. No certificates are used for verifying public keys, so no user setup (and PKI) is required (although the protocol is vulnerable to man-in-the-middle attacks because of this). After generating the session key, all data is encrypted and MACed, and the MAC is stored as a TCP option. Bound to each tpcrypt session is a session id, derived from the key information (master secret). This session id is used to support session caching, allowing hosts that previously communicated to reconnect to each other via a simplified protocol, not requiring public key operations and cipher negotiations. To use session caching, a peer indicates a previous session id (in the SYN packet) from which to generate new key material. The recipient acknowledges (in the SYN-ACK) and no further setup messages are required.

Similarly, for multipath TCP when a new subflow is established, it must authenticate itself to be part of the multipath connection, by supplying the session id of the main flow in the SYN of subflows. This groups subflows into one logical session, where each subflow holds some secret knowledge (the session key).

Tpcrypt will not protect against an active man-in-the-middle attacker (which can inject/modify packets on the fly), because it can force two peers to negotiate two different session keys known to the attacker. It will however detect the presence of a man-in-the-middle if the attacker is not active on all paths, because then the MAC check would fail on at least one subflow. This latter point shows the



---

synergy between tcpcrypt and multipath TCP: tcpcrypt's integrity properties enables authentication for the multipath's subflows, and using multiple paths allows detecting man-in-the-middle attacks on tcpcrypt.

Tcpcrypt is suitable for multipath TCP because it is relatively simple, and built into TCP, making it readily usable for multipath TCP. Alternative solutions such as IPsec, living under TCP, could be applicable though it is a complex solution with non-trivial deployment difficulties e.g. NATs. To protect completely against an active man-in-the-middle attacker would require setting up a public key infrastructure and would result in a complex, difficult to adopt solution. On the other hand, if a multipath TCP session doesn't need tcpcrypt's security, then it can be disabled by selecting the null cipher. Thus, tcpcrypt is a flexible and robust solution applicable to authenticating multipath subflows.

## 3.2 Accountability Framework

One of the main concepts within the Trilogi Architecture is that of shared responsibility. This is most clearly seen in the accountability framework which allows all actors within the network to be held to account for any congestion they cause or allow to be caused. Within the overall Trilogi architecture, the accountability framework sits within the resource plane (although it may influence what happens at the Transport Services level).

Accountability is best assessed at the attachment of a customer to the network from which it gets its connectivity or at the inter-connect between peering networks. At such points there is a direct contractual relationship between the main stakeholders involved and this is therefore a classic tussle boundary. In our earlier work [Trilogi08-D2] analysing the lessons that can be learnt from efforts at applying the design for tussle principle we identified the need to isolate tussles by modularising designs along tussle boundaries and providing simple open interfaces at the boundary.

Currently many ISPs recognise that this point of attachment can be used for more than just pure billing purposes and a number of ISPs already tie their customers to a Fair Usage Policy designed to try and protect the Quality of Experience of the majority of customers from the selfish actions of a minority. In the context of congestion transparency these Fair Usage Policies would be re-written in terms of the congestion – either the rate at which that congestion can be caused or the volume of congested bytes that a user is allowed to send in a billing period.

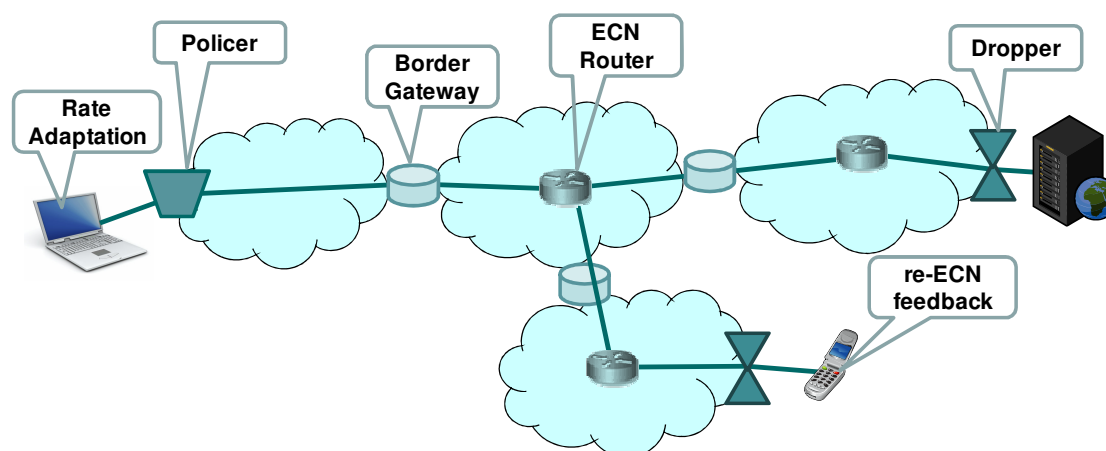
At the inter-connect between different networks the precise nature of the contract may vary: some contracts are mutual – network A promises network B that the same volume of data will flow in each direction across the link; some contracts are supplier-customer – Network A will pay Network B €x per Mbps for traffic at the 95<sup>th</sup> percentile of the peak rate crossing the interconnect. In the context of Congestion Accountability the first case will be replaced by contracts promising to balance the volume of congestion crossing in each direction and the second with contracts where the total volume of congestion is the basis for any charging.

Congestion is the impact one user's traffic cause others (and vice-versa) or, to put it another way, it is what happens when too much traffic meets too little capacity (or even too much pooled traffic meets too little pooled capacity). Congestion volume is simply the congestion integrated over time and over all data; it is easy to measure; all the congestion a user causes is counted, regardless of how many flows are created, the route taken by the traffic, the activity factor of the applications etc. The "user" here means the economic entity: the legal entity accountable for (and paying for) the congestion it causes; this may or may not be the end user.

The elements of the accountability framework are listed here and described in more detail in following sections:

1. **congestion markings.** A link in the network indicates that it is congested by marking congestion information in each packet, as necessary. (Apart from this, the framework avoids locating mechanisms at network resources, and especially if there is no need for network elements to know how much resource a user wants.) This capability has already been standardised as ECN [Ramakrishnan01] and is implemented on many routers, although in general not 'switched on'. Since congestion information is in each packet, this makes it easy to count congestion volume.
2. a new mechanism that gives the ability of any point in the network visibility of the **rest of path congestion**, i.e. congestion between a particular network node and the receiver. Our proposed mechanism is **re-ECN**. Such visibility enables any network node to shape traffic based on congestion experienced across the resource pool by the traffic in question. It thus enables any network node to be given responsibility for the traffic that it forwards, so as well as sources being "accountable" for the traffic they send, an ISP can be accountable for the traffic it sends into another ISP.

3. an enforcement point at the network ingress where the ‘source customer’ attaches. The purpose of this **policer** is to catch those trying to cause more congestion than they’re allowed under their contract [Jacquet08]. Note that a policer is only needed at the ‘first’ ingress; subsequent networks don’t require a policer.
4. an enforcement point at the network egress where the ‘destination customer’ attaches. The purpose of this **dropper** is to catch those trying to cheat by under-declaring the congestion that their traffic causes. Note that a dropper is only needed at the ‘last’ egress; earlier networks don’t require a dropper (although they may choose to implement one at a border gateway).
5. **border gateways**, which audit (count) the congestion volume in each direction between two adjacent ISPs; this is a bulk measurement (not per flow). There might be inter-ISP contracts, similar to those today for bandwidth.
6. **rate control** by the end systems. This is independent from the rest of the accountability framework; the only requirement is that the end-to-end congestion level is declared. Future work will study rate control that takes advantage of the accountability framework to achieve true end-to-end quality of service.



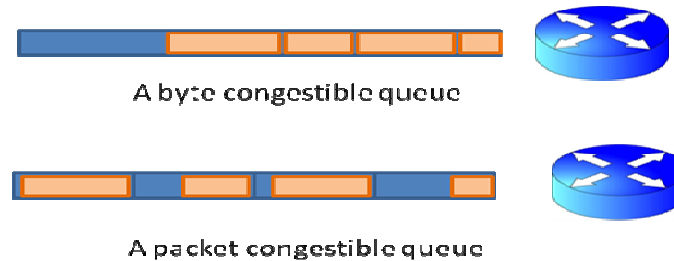
**Figure 7: The six elements of an accountability framework**

### 3.2.1 Congestion notification

As mentioned above, ECN already provides an explicit binary congestion notification through routers marking packets that they would otherwise have dropped. By taking an exponentially weighted moving average of the number of marked packets as a fraction of the total number of packets, you can get an estimate for the level of congestion,  $p$ , used in the classic TCP throughput formula [Padhye98] and in other equation based rate controllers. Thus far within the Trilogy project we have made the assumption that ECN is sufficiently accurate for our accountability framework (a view that is supported by [Gibbens99]). ECN codepoints are already available within the IPv4 header and many routers are already capable of using it. A multi-bit signal may offer some advantages such as faster flow starts, greater responsiveness to congestion and more accurate accounting for congestion. However work on such a signalling system is still in the very early stages.

ECN is only concerned with congestion of router queues. This is because when it was created this form of congestion was the most significant in terms of controlling the rate at which sources could send data. Since then other resources have grown in significance. A mobile device will be most concerned about the impact of any transmission on its battery life and on the level of congestion existing in the wireless access medium. Most networks now incorporate a number of middleboxes performing such tasks as NAT, firewalling or VPN. These middleboxes will often have finite state resources that can become congested. Furthermore the nature of the congestion depends on the nature

of the resource. Most router queues are byte congested – the number of bytes in the queue determines how full it is, but other queues may use a packet-slotted memory and thus the number of packets in the queue will determine the congestion level. Middleboxes are most likely flow congested – each flow requires a certain amount of state and thus the more flows there are the greater the congestion level.



**Figure 8: Byte vs. packet congestible queues**

Within the Trilogy project we are looking at the possibility of overloading the meaning of the ECN field to reflect complex resource usage costs [Trilogy08-D5]. However for the sake of simplicity for the rest of this section it will be assumed that byte congestion at router queues is the most important form of congestion and thus we will talk about ECN referring simply to the marking set on IP packets by router queues running the RED algorithm.

### 3.2.2 re-ECN

Re-ECN [Briscoe05] [Briscoe09b] is a new protocol designed to reveal downstream congestion throughout the network. In the context of a shared network, nodes taking control decisions related to the allocation of shared resources need to get sufficient information about the usage of these resources for the allocation to be made efficiently. For a network node, its control decisions have implications on the consumption of network resources downstream from it. It is therefore very important that information be available about the state of the resources downstream of that node.

In today's best-effort Internet, these control decisions happen at sources and at all nodes along the path that decide which packets to forward and in which order. Sources (and destinations) can get all the information they need about the state (congestion and roundtrip) of the data path, because they have full visibility of both the network and transport layer. Although the information is not perfect (it is provided with a delay of one full roundtrip at the source and effectively coded as a binary signal), it is sufficient for the sources to make the right control decisions for the allocation of network resources.

For nodes elsewhere along the path of the network, it is impossible to establish either the congestion level experienced by a flow or the roundtrip of its path. These nodes cannot detect end-to-end path congestion because losses are only detected by gaps in sequence numbers at the transport layer which may not be visible in the network, for instance if encryption is used. Also network nodes cannot detect roundtrips easily because they lack the self-clocking information inherent to the TCP mechanism.

Re-ECN is a simple extension to ECN utilising 1 extra bit to create 3 new codepoints. Like all apparently simple changes it has far reaching effects in the network. Using re-ECN allows any node in the network to know the downstream path congestion for any packet within the network. This is achieved by using an additional bit in the IP header which, in combination with existing ECN codepoints allows a host to declare the level of congestion they expect a packet to experience along its entire path. This is described in greater detail in [Briscoe09c]. Essentially the congestion signals are encoded into the stream of packets using a unary coding. In part this was a prosaic decision based on the way ECN works, but it has useful consequences in that a unary coding is less sensitive to packet re-ordering and can be read simply by monitoring relative proportions of markings.



---

### 3.2.3 Policer

The goals of the policer design were that users should be free to react in any way they like to congestion within a controlled envelope. The policer provides this envelope by sanctioning users that generate too much congestion. This sanction should be sufficient to ensure that the user can't gain from ignoring it and can't do any damage to the network. [Gibbens99] describes a system where directly charging customers for the number of ECN CE marks they cause leads to a global utility maximisation (under certain assumptions). We have taken this concept a stage further – it is well known that users dislike variable charging, preferring instead charging regimes that give a fixed ration for a fixed price. We have shown analytically [Jacquet08] that a modified token bucket can be used as a bulk policer to limit the rate at which a customer can cause congestion within the network. We have also used a simple simulation to prove that this analysis is correct. Our conclusion from this was that a bulk policer gives a strong incentive to the user to self-police their own behaviour and the penalty of not doing so is greater than the gain. This is true for both responsive and unresponsive traffic.

The policer consists of a token bucket that is modified to control the rate of transmission of re-echoed congestion marks. As long as the rate of re-echo marks is lower than the fill rate of the bucket the user suffers no reduction in throughput. Since the bucket has a finite depth the user is able to build up a small “allowance” that can be used to cause brief bursts of congestion. However if the bucket is empty then the user will be sanctioned in some form (most likely though dropping packets).

### 3.2.4 Dropper

The dropper is used to ensure senders honestly respond to reported congestion (by sending as many re-echoed marks as they have seen CE marks) and also to encourage the receiver to accurately report the number of CE marks it has seen. The dropper checks the balance of re-echoed and CE marks for each flow going across it. Re-echo packets have positive value (for simplicity call it +1) and CE packets have negative value (-1). The aim is for any sender to maintain the smallest possible positive balance at the dropper. If there is an excess of CE marks such that the balance goes negative then the Dropper may apply sanctions to the flow, for instance by dropping a proportion of its traffic. We are currently working on a design for the dropper.

Ideally such a dropper should exhibit the following behaviour:

- **Minimal False Hits:** It should introduce minimal false hits for honest flows. In other words a flow that is honest should get through the dropper un-hindered even if there is a sudden increase in congestion.
- **Minimal False Misses:** It should quickly detect and sanction dishonest flows, preferably at the first dishonest packet.
- **Transport Oblivious:** It must not be designed around one particular rate response, such as TCP's, or one particular resource sharing regime such as TCP-friendliness [Floyd03], given an important goal is to give ingress networks the freedom to allow different rate responses and different resource sharing regimes [Gibbens99, Briscoe07]—unilaterally without coordinating with downstream networks.
- **Sufficient Sanction:** It must introduce sufficient loss in goodput so that sources cannot play off losses at the egress dropper against higher allowed throughput at the ingress policer.
- **Manage Memory Exhaustion:** It should be able to counter state exhaustion attacks. For instance, if the dropper uses flow-state, it should not be possible for sources to exhaust its memory capacity by gratuitously sending numerous packets, each with a different flow ID.
- **Identifier Accountability:** It must not be vulnerable to ‘identity whitewashing’, where a transport can label a flow with a new ID more cheaply than paying the cost of continuing to use its current ID [Friedman98].

### 3.2.5 Border gateway

The border gateway is used to enforce contracts between different operators and to catch attempts to subvert the signalling system. In its simplest form it keeps a count of the number of CE marks and re-echo marks that cross the border. The difference between these values gives the volume of congestion that has been passed across the border in that direction. If this is different to the volume of congestion in the other direction then there will likely be a contractual arrangement to pay for this imbalance in congestion. More advanced designs of the gateway might seek to identify persistently negative flows and push back against them by seeking to block their entry into that network. We are currently working on designs for this and are contrasting how such a mechanism might work in situations with strong competition against those with weak competition.

### 3.2.6 Rate controller

One of the key elements of the accountability framework is making a user responsible for the congestion they cause, thus empowering them to exercise greater control over their network usage within the envelope allowed by the policer. In order to achieve this they need a more powerful rate control mechanism. Traditional rate control mechanisms such as TCP mandate a fixed congestion response that is determined solely by path conditions (RTT and congestion). This response will be the same for interactive applications such as web-based IPTV (e.g. YouTube) and for file downloads, even though it might make the streaming video unwatchable. MPTCP (section 3.1) expands this by allowing traffic to be split across multiple paths but it doesn't mandate how the traffic should react to congestion on those paths. We envisage a more flexible system using a weighted congestion controller. Here the user (or the application designer) sets a weight on each flow. Flows with a very low weight will respond very hard to congestion but will be able to recover quickly once the congestion reduces. Flows with a high weight will only respond minimally to congestion and will be able to get up to speed rapidly. This is illustrated in the following diagram contrasting the behaviour of TCP and a weighted congestion controller for interactive and background traffic.

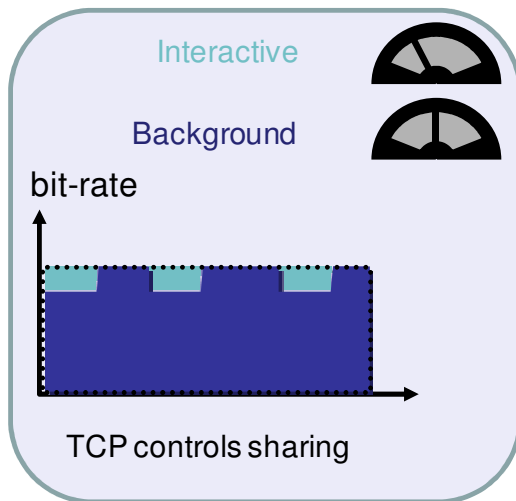


Figure 9: Un-weighted congestion controller

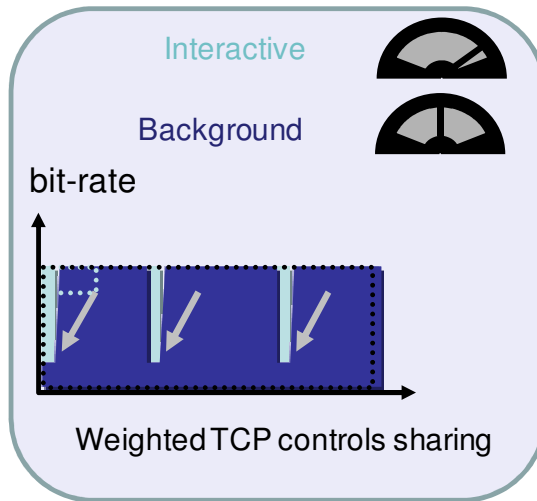


Figure 10: Weighted congestion controller

### 3.2.7 Security of accountability framework

The accountability framework has been designed with security very much in mind. However much of this security comes from an assumption that key players in the network will act in a rational fashion (or at least will act with their own self-interest at heart). A rogue network could create artificial congestion to off-load onto other networks hoping in turn to make their customers suffer (or just aiming to make money out of those networks). However in a competitive market such a strategy won't



---

work for long because the networks being impacted will simply cease to peer with the rogue network. Such application of market forces to solve a problem is at the heart of the Trilogy Project.

The beauty of any system built on congestion transparency is the very fact that it is transparent – every actor in the network is eventually forced to be honest in their declarations of congestion and in their forwarding of congestion markings. There is ongoing debate within the project as to whether congestion marks will represent a flat price, with more marks being generated by more expensive networks (e.g. trans-Atlantic links) or whether the actual price will vary from region to region. The former seems more likely, but whichever it is, market forces will eventually act to keep the charges as near to the true costs as possible.

One of the biggest weaknesses of re-ECN is its susceptibility to source-address spoofing. Malicious hosts could in theory hijack another user's identity and artificially inflate the apparent congestion they are seeing, thus leading to an indirect denial of funds attack. However re-ECN is no more susceptible to this kind of attack than any existing protocol and at least with re-ECN there is the chance of being able to follow an audit trail back to the origin of the fake congestion. Where the dropper is deployed at the egress of a network, source-address spoofing can lead to a flow receiving a greater drop level than it deserves because it may appear to have run into debit due to faked congestion. In such cases the source will see a sudden increase in drop and may choose to abandon the connection. In order to prevent source address spoofing, the Trilogy project is also working on a Source-Address Validation solution to complete the Trilogy architecture. See [Nordmark09] and [Kukec09] for more details.

### 3.3 Multipath Routing

Today's inter-domain routing (to a lesser degree intra-domain routing) is intrinsically single path, i.e. although a typical BGP router learns multiple paths towards a given prefix it is restricted to using only a single path towards it. Using our design principles as a basis, single path routing is generally not a good match for the resource pooling principle. Multipath routing techniques on the other hand enable routers to use different possible paths towards a particular destination according to certain restrictions. Since several next hops for the same destination prefix will be installed in the forwarding table, all of them can be used at the same time, this way increasing the potentially used resource pool. Although multipath routing has a lot of interesting properties, in the current Internet, the required multipath routing support is far from universal.

There are already some, usually underspecified and fairly restrictive, deployed alternatives in order to support the simultaneous usage of multiple paths to reach a certain destination. The best known solutions are the ones being used only within the domain of a particular provider (intra-domain routing) since traffic can be conveniently controlled and directed while all the routing devices are under a single management entity and the multipath solution is typically common throughout the domain. However, these solutions are not directly applicable to the inter-domain routing framework since there are other important factors beyond the technical ones that must be considered, which are mainly related to policy and economic constraints. The following sub-sections elaborate on our multipath routing work both within a single administrative domain as well as inter-domain. While this description might already hint at how such routing is better aligned with our design principles than existing approaches, section 4 will discuss the analysis against the design principles in more detail. These proposals are the subject of on-going research, so liable to change.

#### 3.3.1 Intra-domain multipath routing protocols

Currently, link state routing protocols such as OSPF or IS-IS only use equal cost best paths<sup>4</sup> (Equal Cost Multipath, ECMP) to forward IP packets throughout a domain. The optimality of sub-paths ensures the consistency of hop-by-hop forwarding (loop-freeness property), although paths, calculated using Dijkstra's algorithm, are recursively composed. Depending on the link metric, the diversity of this set of loop-free paths may be insufficient using only ECMP. Therefore, the potential benefits of multipath such as load balancing and fast rerouting are limited and so a more aggressive multipath routing technique might be necessary – a multipath routing approach designed to achieve resource pooling within a domain. To put it in more technical terms, the specific goal is to propose a routing protocol that is able to utilise a larger set of forwarding paths than ECMP.

We do not wish to introduce additional control protocols to achieve this which would be against the project's goal of keeping the control waist in shape. We need to work with the installed base in order to pave a realistic deployment path. We achieve this by building on existing intra-domain routing protocols. Specifically, we propose the following steps:

- Using an unmodified link state control plane to obtain topological information.
- Using a multipath computation algorithm, instead of Dijkstra, to compute candidate next hops.
- Using a loop-free rule to select valid next hops among the set of candidate next hops.

The first step is rather a practical requirement than an active step that needs to be performed. As such, the link state control plane used can be e.g. the one of OSPF or the one of IS-IS. Important for the

---

<sup>4</sup> Equal cost paths are equally good using a particular metric (hop count, delay, price, etc.) and all of them can be used without further degradation. When unequal cost paths are considered, some of them are worse from the used metric viewpoint (more hops, more delay, more expensive), but they are still used to perform multipath routing due advantages outside this simple metric system such as increased resilience or load balancing.



---

overall scheme to work is that the control plane information provided by a link state protocol is available for the second step, the multipath computation.

The multipath computation algorithm needs to generate a set of sub-optimal paths in addition to the shortest path found by the traditional Dijkstra algorithm. One of the challenges for such an algorithm is scalability in terms of time complexity. Indeed, existing proposals have a time complexity which depends on the number of neighbours. The algorithms we propose avoid this issue while also guaranteeing that the number of candidate next hops is greater than two for each destination [Mérindol09a] [Mérindol09b]. The results from evaluations that we have performed on a variety of topologies suggest that the path diversity achieved with our proposals is approximately the same as the one obtained using consecutive Dijkstra computations (one per neighbour), but allows to achieve a significant time complexity gain. In fact, our multipath computation algorithms do not need to perform consecutive Dijkstra computations. They take advantage of the path diversity explored in a single Dijkstra run to perform specific path compositions<sup>5</sup>. Path diversity is a fundamental piece in the resource pooling puzzle.

To ensure that all the paths are loop-free we apply some well known rules [Vutukury99]. These Loop-Free Invariants (LFI) allow each router R to forward its traffic to a destination D via each neighbour V whose best cost towards D is strictly lower than the best one of R. LFI is easy to implement if each router knows a set of candidate next hops (a set of potential neighbours V) towards any destination.

With our propositions the loop-free rule is locally verified (without exchanging any signalling message) and does not involve all routers, i.e. no additional functionality added to the “control waist”. The deployment can therefore be incremental because we minimise the changes in the control plane of each router. In summary, our propositions can be incrementally integrated in OSPF or IS-IS by replacing Dijkstra by our path computation algorithms and using a loop-free rule such as LFI.

### 3.3.2 Inter-domain multipath routing

Inter-domain multipath routing, as compared to intra-domain routing, is clearly a harder problem to solve as in principle multiple organisations need to agree on the deployment and additional economic considerations make routing operations more complex. Some proposals have already been made in the inter-domain context but due to different reasons, these existing proposals for multipath BGP have still not been introduced as real alternatives. Some of them require major changes in routing operations or even require the deployment of a new inter-domain routing protocol, others are restricted to a very limited set of possible routes and are potentially not even officially specified/standardised such as the Cisco implementation of multipath BGP. To demonstrate that these limitations can be overcome in a way that is compatible with our design principles (again mainly the resource pooling principle), we are designing particular proposals based on the following motivations and assumptions for an early adoption:

1. Change BGP semantics as little as possible.
2. Change BGP routers as little as possible.
3. Be interoperable with current BGP routers.
4. Provide more path diversity than exists today.

We currently explore two potential proposals that will be briefly outlined in this section. In addition to the mentioned motivations, it is worth noting that any solution should comply with the peering/transit Internet model based on economic considerations (see [Gao00]). The rationale for this model is to realise that in most cases a site only carries traffic to or from a neighbour as a result of being paid for

---

<sup>5</sup> For reasons of brevity and clarity we omit the details of the path composition process which is explained in great depth in [Mérindol09a].

this, or because of an agreement exists in which both parties obtain similar benefit (peering). This results in the requirement to enforce two major restrictions:

- Egress route filtering restrictions: customer ASs should advertise their own prefixes and the prefixes of any customers, but they should never advertise prefixes received from other providers (i.e. an AS should never advertise to its peers more than its own prefixes and those of its customers). In this way, a site does not offer itself to carry traffic for a destination belonging to a site for which it is not going to obtain direct profit.
- Preferences in route selection: routers should prefer customer links over peering links because sending and receiving traffic over customer links generates revenue, and peering over provider links, because peering links at least do not cost them money. According to this, the multipath route selection process can aggregate routes from many different customer links; or many peering links; or many provider links; but it can never mix links associated to different relationship types. Note that the administrator of a site may have specific preferences amongst routes received from its neighbours with the same relationship type, because of economic reasons, traffic engineering, etc.

As a result of the peering/transit model, paths in the Internet can start going ‘up’ from the originating site to a provider, and up to another provider, many times until it reaches a peering relationship, and then descend to a customer of this site, descend again, many times until it reaches the destination. Since it is impossible to find paths in which descending from a site to a customer and ascending again, or paths in which a peering link is followed by an ascending turn to a provider, it is said that Internet is ‘valley-free’ [Gao00] as a result of the application of the peering/transit model.

The ‘valley-free’ model suggests that a loop in the advertising process (i.e. a route advertised to a site that is already part of the AS PATH) can only occur for a route received by a provider. This is because a customer or peer of a site (S), cannot advertise a route that has been previously advertised by S, according to the restrictions stated above. The valley-free condition also assures that a route containing S that is received from a provider P1(S) was advertised by S to another provider. Since S only announces its own prefixes or customer prefixes to its providers, the prefixes received by any provider, whose selection would result in a loop, are its own prefix or customer prefixes. Note that these routes would never be selected because either the destination is already in the site, or because it always prefers customer links to provider links.

Consequently, although there is a specific mechanism in BGP for detecting loops in the routes, the application of the peer/transit model by itself would be enough to assure that loops never occur. Of course, loop prevention mechanisms must exist in order to cope with routing instabilities, configuration errors, etc. We can extend this reasoning to the multipath case to state that, if any multipath BGP strategy complies with the peering/transit model, as requested before, the aggregation of routes with equal condition (just customer routes; if not, just peering routes; and if not just provider-received routes) will not result in route discarding due to loop prevention in the steady state for well-configured networks.

However, any multipath BGP mechanism must provide a real loop prevention mechanism to cope with e.g. transient conditions and configuration errors, and regular loop detection techniques available in BGP are not sufficient as we will show. In this project we are exploring two proposals that share some mechanisms, such as part of the route selection approach, and differ in others, such as the loop prevention mechanism.

### ***Multipath selection***

The path selection process for multipath BGP should take as a starting point the rules for unipath BGP, deactivating the rules that are used for tie-breaking among similar rules to allow the selection of multiple routes instead of just a single one. Note that the more rules that are deactivated, the larger number of routes with the same preference can be selected for multipath forwarding. However, as mentioned before, only routes that are equivalent for the administrator must be selected, resulting this



---

preference from economic reasons, traffic engineering considerations, or in general any policy that the administrator wants to enforce. So a modified multipath router first applies normal BGP policy criteria and then selects a subset of the received paths for concurrent use. The attributes and rules through which relevant preferences of the administrator are enforced, in the order in which they are applied, are:

- Discard routes with lower LOCAL\_PREF. This rule enforces any specific wish of the administrator, and is the rule used to assure that only routes received from customers are selected; or if no routes from customers exist, only routes received from peers; or if none of the previous exist, routes received from providers.
- Discard routes with higher MED. This rule is used to fulfil the wishes of the customers in order to implement ‘cold potato’ routing so that customers’ costs in terms of transit cost are reduced.
- Discard lowest ORIGIN. This rule is used in some cases as a traffic-engineering tool. If not, the impact of its application is low, since almost all routes should have equal ORIGIN attribute.
- Discard iBGP routes if eBGP routes exist. It is used to deploy hot-potato routing, which may be relevant to reduce internal transit costs. In addition, it also eliminates internal loops in route propagation. When applied, routers receiving a route from an external neighbour use only external neighbours, so internal loops never occur. Routers not receiving a route from an external neighbour selects the router inside the AS that will send the packet out of the AS.
- Discard routes with higher cost to NEXT\_HOP. This is also used to enforce hot-potato routing. However, some relaxation on this rule can be introduced, provided that prevention of loops in intra-domain forwarding is achieved by means such as some kind of tunnelling like MPLS.

The rest of the rules (selecting route received from router with minimum loop-back address, etc.) are provided to ensure uniqueness in the result, so they can be removed for multipath routing.

Therefore, a modified router first applies normal BGP policy criteria and then selects a subset of the received paths for concurrent use. Note that multiple paths mainly come from the possibility of ignoring AS\_PATH length (although some conditions on this length could be established for accepting a route), and from accepting routes with different NEXT\_HOP distances.

#### ***LP-BGP: Loop-freeness in multi-path BGP through propagating the longest used path***

In this particular proposal [Beijnum09a], after obtaining the different paths that will be installed in the forwarding table for the same destination prefix, the path with longest AS\_PATH length to upstream ASs will be disseminated to neighbouring routers where allowed by policy. It is important to note that “longest path” here does not refer to the ultimately longest path found in the Routing Information Base but to the longest path installed in the FIB, i.e. used to forward packets. There can be limitations on the path length of this “longest path” such as a maximum number of hops longer than the shortest path. This could also mean that the longest path is actually the shortest path, which would restrict the number of path a router could use concurrently to all paths that have the same LOCAL\_PREF and the same path length. Although disseminating a path that has a larger number of ASs in its AS\_PATH seems counterintuitive, it has the property of allowing the router to use all paths with a smaller or equal AS\_PATH length without risking loops.

However, this change has the implication that there is no longer a one-to-one relationship between the paths that packets follow through the network and the path that is advertised in BGP. The resulting obfuscation of the network's topology as seen by observers at the edge can either be considered harmful, for those who want to study networks or apply policy based on the presence of certain intermediate domains, or useful, for those intent on hiding the inner workings of their network. The multipath BGP modifications allow individual ASs to deploy multipath BGP and gain its benefits

---

without coordination with other ASs. Hence, as an individual BGP router locally balances traffic over multiple paths, changes to BGP semantics are unnecessary.

Under normal circumstances, the BGP AS PATH attribute guarantees loop-freeness. Since the changes allow BGP to use multiple paths concurrently, but only a single path is disseminated to neighbouring ASs, checking the AS PATH for the occurrence of the local AS number is no longer sufficient to avoid loops. Instead, the Vutukury/Garcia-Luna-Aceves Loop-free Invariant (LFI) [Vutukury99] conditions are used to guarantee loop-freeness, which is what is described above.

#### ***MpASS: Multi-path BGP with AS sets***

The main idea behind MpASS is to include all the AS numbers resulting from the union of the AS PATH attributes of the routes aggregated into the single advertised AS PATH. In particular, the AS PATH is obtained by concatenating an AS SEQUENCE structure containing the AS PATH corresponding to the route that the BGP router would select from applying BGP unipath selection rules, and an AS SET structure that includes all the AS numbers of the rest of the routes, and the AS number of the site. This particular construction mechanism assures that all AS numbers are included and the length of the AS PATH structure as defined for the AS PATH length comparison rule [Rekhter06], is equal to the length of the AS PATH of the best route plus 1 (as it would occur for unipath BGP routers). In this way, when a legacy (unipath) router applies the rule of discarding routes with larger AS PATH length, this multi-path route is not penalised compared to the unipath route that it would have generated.

Loop prevention is enforced by the check performed by regular unipath BGP and it is not necessary to define any additional mechanism or particular condition. In other words it is sufficient to discard routes that contain the AS number of the site of the router receiving the advertisement. An additional characteristic is that the inclusion of all the AS numbers of the sites that may be traversed by a packet sent to the destination allows the application of policies based on the particular AS traversed when selecting a route. Legacy BGP routers receive a route that is indistinguishable to a regular BGP route, and if they select it, packets may benefit from the multiple available paths.

#### ***Proposal comparison***

These two mechanisms mainly differ in the way routes are selected and how loop prevention is enforced. LP-BGP, has the potential to reduce the number of BGP updates propagated to neighbouring routers, as updates for shorter paths do not influence path selection and are not propagated to neighbouring routers. However, with longer paths there is more potential for failures, so the inclusion of long paths (in the set of paths that a multipath router uses) may expose it to more updates compared to the situation where only short paths are used. When propagating just the longest path, BGP no longer matches the path followed by all packets. MpASS allows the selection of routes with any AS PATH length, since loop prevention relies on transporting the complete list of traversed AS numbers.

One difference among them is that LP-BGP may propagate a route with an AS PATH larger than the best of the aggregated routes, so that the result of a multipath aggregation may be a route less attractive to other BGP routers (presenting longer paths to customers may put service providers at a commercial disadvantage). Still, propagating the longest path has robust loop detection properties and operators may limit acceptable path lengths at their discretion, so the second disadvantage is relatively minor (they could require for instance all best routes to be equal length).

On the other hand, MpASS may suffer from excessive update frequency, since each time a new path is aggregated in a router, a new update must be propagated to all other routers receiving this route, to ensure that loop prevention holds (note that in the unipath case, BGP only propagates a route if the newly received update improves the previous route, while in this case many routes may be gradually added to the forwarding route set). This problem can be relieved by setting a rate limit to the aggregation process.



---

### 3.3.3 Intra-domain and inter-domain path diversity combination

If both intra- and inter-domain routing provide multiple forwarding paths, then it is possible to combine these sets of paths in order to increase the network reliability. The goal is to take advantage of the existing path diversity at these two levels. However, there are some implications which we describe with reference to the example given in Figure 11. We consider a given destination prefix P advertised by AS3. The traffic entering in AS1 via R1 (the ingress ASBR router belonging to AS1) can be forwarded through several paths.

On the one hand, at the inter-domain level, to take advantage of the inter-domain path diversity, an incremental solution is to use the existing redundant connectivity between adjacent ASs (AS1 and AS2 are connected through routers R2 and R3). Such redundant connectivity (or multiconnectivity) between autonomous systems (ASs) is important for inter-domain traffic engineering and fast recovery in case of failures. However, the redundancy of ASs peering links has not been quantitatively studied, mainly due to the difficulty of obtaining relevant data.

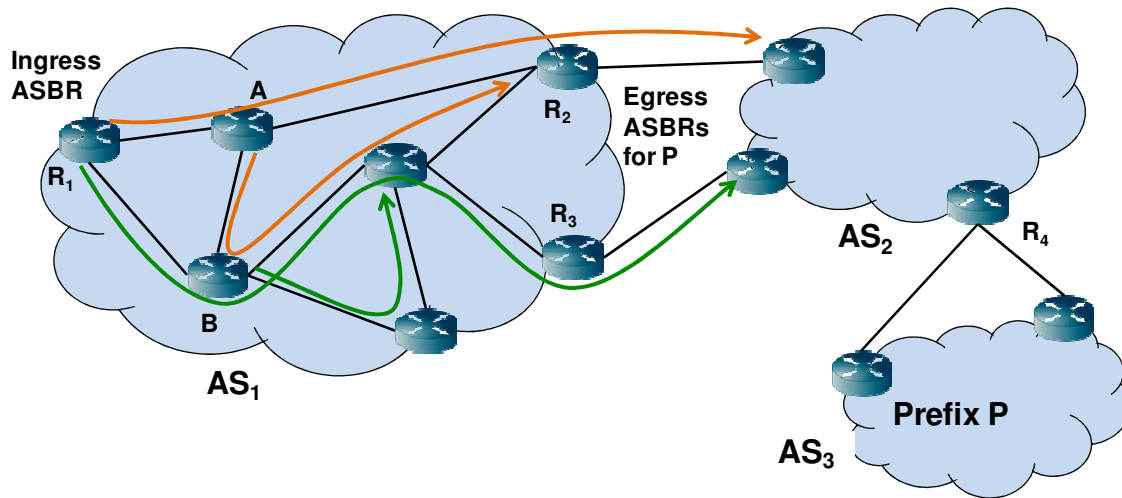
We used the minfo multicast monitoring tool to provide useful data about the Internet topology and such links in particular. Our analysis relies on more than four years of daily queries to about ten thousand routers divided in more than two hundred ASs [Mrinfo]. We demonstrate that peering links between ASs are frequently redundant. In particular, as a lower bound, our analysis shows that more than half of the studied ASs pairs are connected through multiple physical links. We then refine our analysis by considering the different types of ASs (Tier-1, Transit, Stub) and their business relationship (peer-to-peer, customer-to-provider, provider-to-customer). A particular result of our analysis is that roughly 75% of the peer-to-peer relationships between adjacent Tier-1 ASs are redundant. Moreover, in most of the cases, multiconnectivity between Tier-1 ASs is ensured by several AS border routers.

These results tend to demonstrate two interesting properties to deploy a simple scheme for inter-domain multipath routing.

On the one hand, it allows ASs to provide a simple recovery mechanism in case of inter-domain link or border router failures. On the other hand, considering a relaxation of the BGP tie-break decision process, the existence of multiple links between a given pair of BGP neighbour ASs allows to increase the potential throughput for the traffic forwarded via this AS\_PATH sequence. Indeed, with an appropriate traffic engineering mechanism at the ingress domain, it is possible to balance the load through this set of multiple connections.

Our measurements suggest that it may be sufficient to use the existing diversity between neighbour ASs to ensure an inter-domain multipath routing protocol able to provide a simple scheme for resource pooling. Indeed, this specific kind of connectivity redundancy can provide the resource necessary for traffic engineering objectives and allows us to deploy an incremental inter-domain multipath routing protocol. In addition, an inter-domain multipath routing protocol can take benefit of the internal diversity of each ASs used to reach the destination. In particular, it is possible for an ingress edge router to use several internal paths (inside the domain it belongs) towards several egress edge routers connected to the next hop AS. For instance, the ingress edge router can select all paths having an IGP cost in a given range to reach the set of possible egress edge routers allowing to reach a given prefix.

Again, the approach is to relax the BGP tie-break decision rule in the decision process. Using the example in the figure, assuming that R1 is able to select R2 and R3 as egress points for P into AS2, AS1 can use its redundant physical links with AS2. We consider, in this example, that the AS\_PATH “AS1-AS2-AS3” is the shortest path towards P, and that R1 has selected R2 and R3 as egress ASBRs towards P because their respective costs are within the same range (by “cost”, we mean the IGP costs between R1 and R2 and R3, plus the metric attributed to their respective inter-domain links). A necessary condition to enable this feature is to advertise the existing multiple paths in BGP [Walton09], and redistribute them in the intra-domain with iBGP



**Figure 11: A possible combination between intra- and inter-domain multipath routing**

On the other hand, at the intra-domain level, if AS1 enables multipath in its IGP, the router denoted A may use two alternate paths to reach R2 (and similarly router B can use two paths to reach R3). An alternative might be for R1 to use multiple tunnels (e.g. MPLS or GRE) to position multiple paths towards R2 and R3.

The intra-domain and inter-domain path diversity may be combined. This depends on the loop-free rule used in AS1 and the iBGP external route redistribution.



---

## 4 Analysis of the architecture against the Design principles

The Trilogy Design Principles are intended to help engineers and researchers ensure that their designs follow the principles of Design for Tussle. As with many design principles the idea is not to blindly follow them without thought or understanding. These are first and foremost guidelines and not rules. Consequently not all design principles will apply to all situations. This is most obvious with the Fuzzy Ends principle – this is a limited relaxation of one of the original design principles of the Internet. We are not saying that *all* end-host functions must be delegated into nodes in the network; rather we are saying that such delegation may be necessary in some circumstances and will still lead to a system that is designed for tussle. This section examines each of the 3 main work blocks and shows how their design was influenced by the Trilogy Design Principles. The following section highlights how they all fit together into a coordinated architecture largely because they follow these Design Principles.

### 4.1 Multipath TCP

The design of the multipath TCP protocol was heavily influenced by the Resource Pooling Design Principle. To a lesser extent it also relies on the Information Exposure and Separation of Policy and Mechanism Design Principles. These are explored in more detail.

#### 4.1.1 Resource pooling using multipath transport

Multipath TCP is a resource pooling mechanism which splits data across multiple subflows and ensures reliable delivery across this set of subflows. Resource pooling in this manner allows the load to be relocated or spread over several resources. In the case of Multipath TCP, this load is data and the resource is links. This increases resource utilisation, increases robustness (so long as at least parts of the paths are disjoint), and is better placed to handle sudden increases in demand for bandwidth.

Multipath TCP is more responsive to failure and other path changes. The failure of a path can be regarded as an extreme case of congestion. When multiple paths are used simultaneously, Multipath TCP will react to failures by diverting the traffic through paths that are still working and have available capacity. Redundant information can be used across the multiple paths so that even if packets in flight are lost due to path failure, enough information is delivered to the destination via the remaining paths. Old and new paths can be used simultaneously and the traffic distribution can adapt to the available rate for each path, making a smooth handover without suddenly dumping traffic onto a new path and causing congestion.

The improved response to path changes is a very important motivation for both end users and network operators. One caveat within the design principle is that resource pooling methods should not conflict with each other. The MPTCP protocol does not explicitly avoid this, and so in theory conflicts could exist. If a node does not have full knowledge of paths available to it, it may make a suboptimal choice of scheduling. If there were intermediate MPTCP implementations (e.g. proxies) these could conflict, although strict application of the fuzzy end-to-end principle should lead to the endpoint being aware of the presence of such implementations in the path.

If both end-points understand the necessary extensions to support the two-ended variant of MPTCP then the connection will be able to exploit all available paths and will achieve good resource pooling. However if one of the endpoints is unmodified then this resource pooling is not possible and the connection will behave as a standard TCP connection. In such cases if the sending endpoint understands the one-ended variant of MPTCP then this can be used to achieve path diversity and resource pooling.

---

#### **4.1.2 Information exposure – making better use of available information**

The information exposure design principle refers primarily to transparency in terms of resource usage. In other words, when a node makes use of network resources, it should also provide control information by which this usage can be monitored, and through which an efficient allocation of scarce resources can be achieved. Multipath TCP monitors the congestion signals on each individual subflow, in order to respond appropriately to resource usage/congestion by shifting load between the subflows. As such it is a beneficiary of the Information Exposure principle. Separating

#### **4.1.3 Separation of policy from mechanism**

The separation of policy and mechanism has been integral to MPTCP from the beginning, as it is possible to specify the protocol without having to specify how the end host decides which path to use. This allows MPTCP to be deployed in various situations, including full resource pooling or as a backup path, depending on an end host's needs and path characteristics. This policy is entirely separate (and local to an end host) from the mechanism (which is the protocol, standardised across all parties).

MPTCP does not mandate where a policy can originate, and its separation from the mechanism allows it to come from any number of places. Although policy is typically automatic on an end host, possibly based around a simple user preference (e.g. maximum throughput or minimise monetary cost), an application could express its own preferences for traffic to varying degrees of granularity (as discussed in Section 6.3).

#### **4.1.4 Fuzzy ends principle**

The current MPTCP protocol proposals are designed for use by end hosts, however the architecture proposed is sufficiently extensible to allow the development of "proxy MPTCP" boxes that could be placed within the network to provide multipath benefits without the need of endpoints to be multihomed themselves. Such an approach could employ a 'path ID' in the form of a tag on a packet that can be used to indicate a route preference to an MPTCP proxy. Such path IDs could also be used in the context of multipath routing to allow end-hosts to reveal their preferences to the network.

### **4.2 Accountability Framework**

At its heart the accountability framework is all about Information Exposure. More specifically it is based on the re-feedback protocol which achieves congestion transparency throughout the network. By ensuring congestion is visible at the layer that needs to know about it you assist with the Separation of Policy and Mechanism.

#### **4.2.1 Information exposure - congestion transparency using re-ECN**

Congestion marking by network nodes ensures that information on resource scarcity is available to the end-systems. This information is carried in the actual packet header and is aggregated along the data path so that end-systems get a full picture of the state of congestion of the path. The re-ECN mechanism ensures that this information about congestion on the path ahead is available at any node in the network. This is particularly important for policing, which should ideally be done at attachment points between stakeholders and should impose sanctions that are visible in-band (such as drop, delay or marking):

- at the attachment of a customer to its network provider
- at connections between different network providers



---

The framework developed so far has focused on dealing specifically with congestion of bandwidth bottlenecks. Accountability for other congestible resources would require separate signalling in the same vein as is done for bandwidth congestion (in-band, transparent and aggregatable).

The Information Exposure principle has significant bearing on the nature of the sanction applied when the congestion allowance is running out for a user. Indeed, the congestion allowance itself can be seen as a congestible resource. Ideally, what would matter with that respect is that the user be given an explicit signal of the (existence and) intensity of the sanction, and that this signal be visible in-band at a minimum to the end-systems and potentially to intermediate nodes. This allows the user to adjust their behaviour to optimise their network experience within the constraints imposed by the congestion allowance.

#### **4.2.2 Separating policy and mechanism**

At first sight, it might seem that we embed policy in a mechanism: congestion accountability seems to define the policy by which network usage should be policed. However, the fundamental architectural statement is for congestion transparency, which only requires the congestion information to be available. This allows for active congestion accountability mechanisms to be deployed such as the congestion policer [Jacquet08], but operators could still choose to ignore that information in their traffic control policy. We expect that competition rather than policy will ensure the transition towards active congestion policing. Congestion transparency ensures a wider spectrum of traffic control policies, and enabling policy choices that actually result in optimal social welfare for the community of users [Kelly98]. Such policy choices might include how to sanction users, which packets and flows to target and whether to use the information only for performance monitoring.

Another point where policy choices seem to be embedded into technical mechanisms is that the congestion signal accumulates on an end-to-end path even though all the links making up the path are not equal; some links may be more congested, others may be more expensive to run or to upgrade. This minimal approach to building the e2e congestion signal is actually all that is needed to ensure separation between policy and mechanism. Indeed, owners of the congested bottlenecks are at liberty to choose how they want to translate bandwidth overload into congestion markings: they can decide the shape of the marking algorithm: whether to use the RED algorithm, which variant to use, how to set the parameters, whether it should be driven by the actual queue in the buffer or by a virtual queue.

Similarly every network operator has the flexibility to make their upgrade decisions according to their own policy. If two links are equally congested but one is more expensive to upgrade, then its owner will wait longer before upgrading because a higher demand is required to warrant the expense. Choosing to control congestion volume is a choice that is as policy-neutral as controlling total volume because it doesn't spell out which rate control mechanism end-users need to use, and it doesn't hardwire assumptions on a link between the nature of the traffic and its desirability.

#### **4.2.3 Fuzzy Ends**

Proxy mechanisms can assist with the deployment of re-ECN. Where users are unable or unwilling to upgrade their network stacks, a re-ECN proxy in their home router could add this functionality for them. This might be provided by the router manufacturer, their ISP or by a 3<sup>rd</sup> party. Such proxies would provide congestion transparency but would have only indirect control over the user.

#### **4.2.4 Resource pooling**

The accountability framework is an essential requirement for resource pooling across multiple providers, however in and of itself it doesn't actually create resource pooling. Consequently this Design Principle is not directly relevant to the accountability framework.

---

## 4.3 Multipath Routing

Like Multipath TCP, Multipath Routing is all about exploiting Resource Pooling.

### 4.3.1 Resource pooling through multipath routing

Multi-path routing implements the resource pooling principle. In single path routing environments resources of equal cost (or even unequal cost if desirable) are not utilised concurrently for a given destination or address range. Multi-path routing pools these resources which is different from the way multi-path TCP pools resources as it happens within the network, unaware and independent of congestion and not relying on multiple address pairs. In particular, multi-path routing:

- pools the capacity of multiple paths. Capacity pooling at the routing layer is statistical and localised through mechanisms such as ECMP rather than based on measurements and an end-to-end view. But the capacity of multiple paths is pooled as for a given destination packets might travel along different paths.
- pools the reliability of the paths being used. If a path fails, the others can still be used. The fraction of traffic that was served by the failed path is distributed across the remaining paths until the failure has been addressed. This ensures service continuity and reliability.

### 4.3.2 Separation of policy and mechanism

The work on multi-path routing is centred round the fundamental properties of routing systems such as loop-freeness. Following this principle, we have not embedded fixed policy mechanisms into the protocol. The policy aspects are outside this work and left as a local decision. Policy in this space includes how traffic is split across a set of paths. This could be done with an algorithm such as ECMP that uses a hash over the 5-tuple to avoid re-ordering at the transport layer. It could be done using a statistical unequal cost hashing function if an operator prefers a number of paths out of the set of the given paths. It could even be achieved by loosely coupling path selection with congestion control. Policy also comprises the decision on which paths should be in a given set towards a destination. Our work has not restricted the set of paths themselves unless it was necessary to ensure e.g. loop-freeness. The mechanisms developed have not restricted the viable paths according to metrics such as path length as that would have restricted the policy freedom of the operator.

### 4.3.3 Information exposure

One large part of the Multipath Routing work is concerned with the architectural split between Locators (required by the Reachability Plane) and Identifiers (part of the Resource Plane). Without the concepts of locators and identifiers it would be impossible to correctly assign usage information to the correct entity. Thus these concepts are fundamental to the Information Exposure Design Principle. In addition to this by revealing more information within the network (as part of the accountability framework) you move more closely to the original aim of self-describing datagrams. With this additional information carried on every packet you allow the network to make more intelligent routing decisions about packets and flows.

### 4.3.4 Fuzzy ends

The Fuzzy Ends Principle is about delegating functionality into the network. Thus it is not relevant to the work we have done on multipath routing which is a function that is wholly inside the network.

---

## 5 Combining the architectural building blocks

### 5.1 Overview

On their own each of the three individual building blocks - multi-path transport, multi-path routing and congestion accountability – are highly useful. In other words, under certain assumptions they fulfil a particular architectural goal without the presence of the other building blocks. In this section we go one step further and explain why a “loose” coupling of the three building blocks can significantly broaden their architectural footprint [Bagnulo09].

#### 5.1.1 Routing

Let us start with routing. Currently for stability, routing operates on timescales substantially larger than transport protocols ( $\gg$  RTT). Traffic engineering for example is not a mechanism that can react dynamically to changes in the quantity of traffic. By giving the routing system additional resources for forwarding (i.e. multi-path routing) the traffic matrix that the network can accommodate increases which makes the network more robust and means resources are used more efficiently. Unfortunately, that does not change the timescales on which the routing system can react to changes of the flow of traffic. For processing simplicity and stability, techniques such as ECMP used with intra-domain multipath, do not take traffic volume or downstream congestion into account. This is sub-optimal locally, and may have a negative effect on networks downstream. Local traffic engineering lacks the end-to-end view required to reach a globally optimal flow of traffic.

#### 5.1.2 Transport

Now, consider congestion control and transport protocols. Today transport protocols like TCP use a single path, and the reaction to network conditions is limited to increasing or decreasing the amount of traffic injected into the network. Multipath transport enables an alternative reaction – shift some of the traffic to another path. This reaction happens at RTT timescales and can address the deficiencies mentioned in the previous paragraph. But of course there need to be multiple paths available, and it is best if the paths are disjoint – otherwise the traffic may still go through the same bottleneck link, in which case using the alternative path makes no difference. So with multiple paths available, a loose coupling of routing and congestion control becomes attractive.

#### 5.1.3 Accountability

Allowing users access to a larger resource pool has a couple of implications for the accountability of the resource usage though. Firstly, if multiple paths are used simultaneously then the users will tend to consume more resources than a single path user. This presents a challenge for an operator, who would like to apportion its costs equitably. Indeed, P2P applications, which download from many peers simultaneously, have resulted in operators adding deep packet inspectors in order to control, albeit crudely, the amount of bandwidth a P2P user gets. Secondly, the concept of multi-path implies that traffic may be sent over non-optimal paths, which may incur some extra cost for the operator. Thirdly, the operator needs to trust that the end users really are shifting their traffic and load balancing appropriately over the different paths – and not for instance acting maliciously. All these factors suggest that adding accountability for resource usage is an important step, as it enables the operator to apportion equitably the extra costs from multi-path, which encourages its deployment.

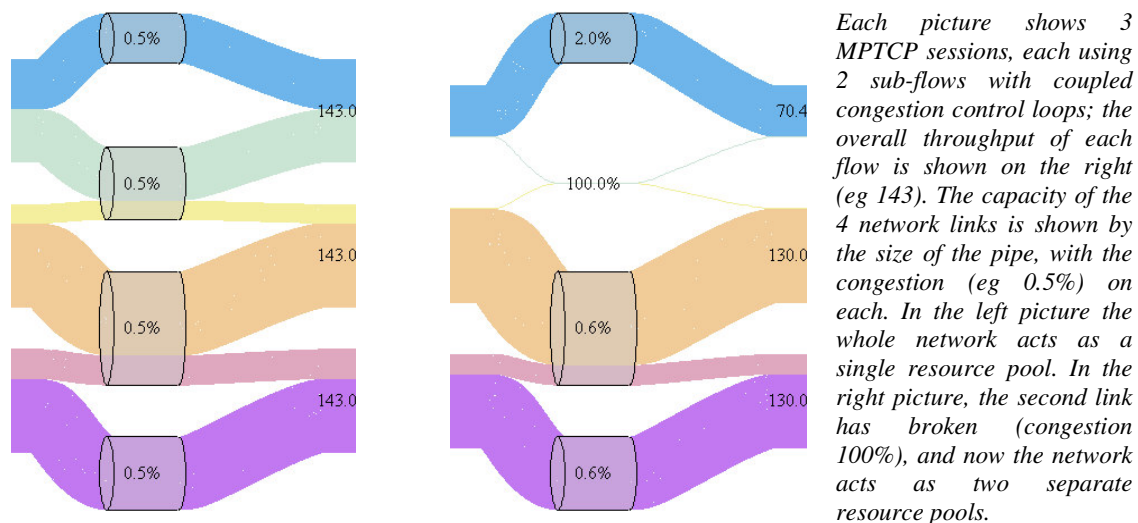
## 5.2 A more detailed consideration

Multipath transport enables traffic to spread adaptively over multiple paths. There are three objectives:

1. increased robustness against component failures;
2. better ability to handle localised surges in traffic;
3. maximise utilisation of the network's resources.

From a global perspective, a congestion control algorithm can be viewed as solving an optimisation problem: to maximise the “social welfare” of all the users, using a distributed mechanism, in a dynamic real-time manner. Ideally the network should behave as a single pooled resource, rather than a series of separate resources. When a link fails in one place, or there's a traffic surge on one part of the network, then ideally the overall impact is dissipated over the whole resource pool (hence “resource pooling”), which thereby brings the benefits of reliability, flexibility and efficiency.

The purpose of a multipath transport is to enable this re-balancing of traffic, by sessions shifting their traffic (or some of it) from one path to a less congested one<sup>6</sup>. However, this global optimisation is an idealised picture. There is an open question about whether the traffic can actually re-balance completely, or whether (for instance) the re-balancing can only take place over a few local links. One can imagine the network as a series of small, partially isolated resource pools, whereas ideally the whole Internet acts as a single resource pool. Overall the resource pooling is likely to be better if more hosts use multipath transport, and if the multipath transport can choose between several paths and these paths are disjoint (i.e. they don't have the same bottleneck link), as illustrated in the Figure below.

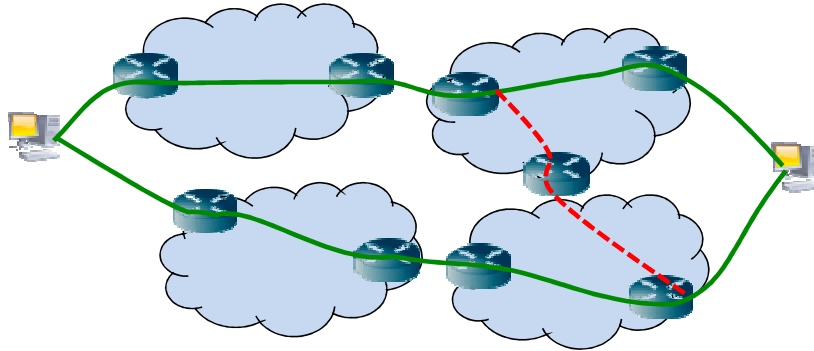


**Figure 12: Network acting as a single resource pool (left) and two resource pools (right)**

The essential reason for multipath routing is to improve the resource pooling. Multipath routing gives more paths for MPTCP to explore and enables MPTCP to operate even when the host is single-homed.

<sup>6</sup> In general we have assumed that traffic from a session is split over several paths simultaneously, with linkage of the congestion control loops of the various sub-flows. This is the aim of MPTCP. However, it should also be possible to adapt differently, eg perhaps an interactive application only uses one path at a time, but can switch as the best path alters. This is for further study.

Consider **Error! Reference source not found.** below. It assumes the end-points are dual-homed on two providers. It shows two distinct paths being used (solid lines). Each path has a different source-destination address pair. When the top path becomes congested then the session can either go faster on the bottom path, and/or a new path (dashed line) is used (it has the same source address but a different destination address).

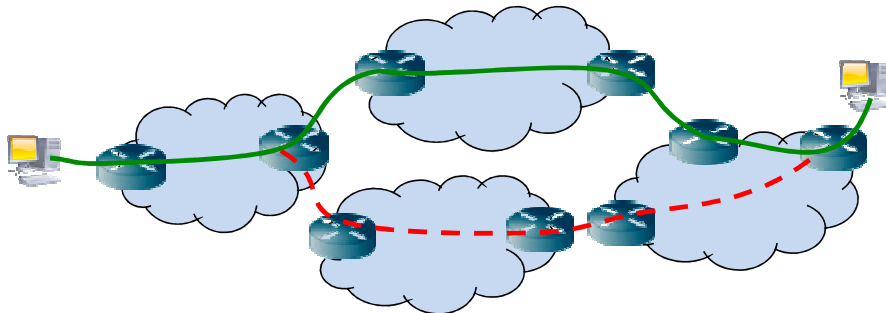


**Figure 13: MPTCP exploits multiple addresses**

This works sufficiently well under two assumptions:

- A. the paths are sufficiently diverse, ideally not sharing a bottleneck
- B. end-host multi-homing becomes the rule rather than the exception

Assumption A is not a first order goal of routing today. Part of our research is to assess just how diverse paths are in practice. Assumption B is a reasonable assumption and a trend that can be observed today, in particular for mobile devices. But a single-homed device can also make use of multi-path transport. Two single homed devices can use partially disjoint paths by influencing the path choice of routers depending on the congestion they experience on a given path. One possibility is that a host has more than one IP address, and so using a different source-destination address pair exercises the dashed path. An alternative option is shown in **Error! Reference source not found.** where the transport protocol adds a path identifier to its packets, and this triggers the router to redirect packets over the path indicated by the dashed red line. This could represent a single-homed customer communicating with a multi-homed provider.



**Figure 14: MPTCP exploits path selection**

The fundamental points are these. The “loose coupling” of routing and congestion control improves the pooling of the network’s resources. The coupling is “loose” in two ways. Firstly, the transport protocol does not need to know explicit path information, which avoids additional complexity issues. The transport protocol simply ‘requests’ that packets follow different paths (whether through different address pairs or through different path identifiers) but cannot ‘demand’ a specific path, nor even ‘demand’ a guarantee that the paths are disjoint. Secondly, the routing system doesn’t need to know

---

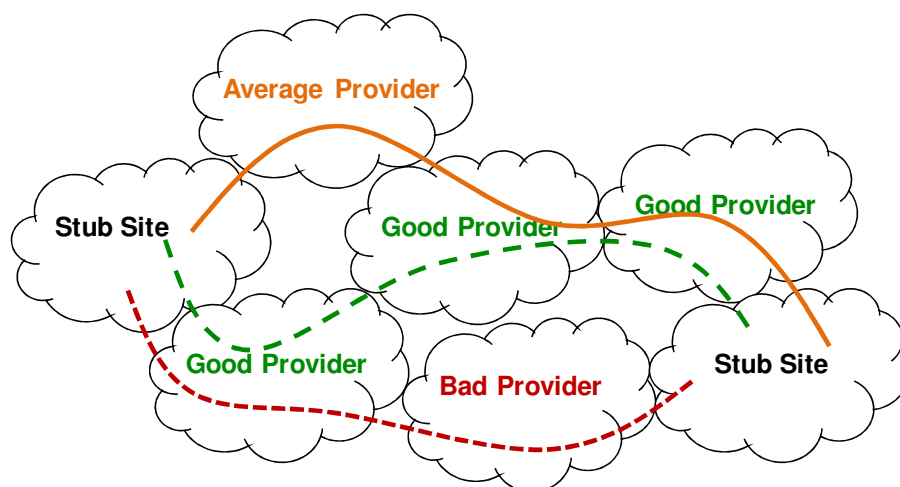
explicit congestion information, i.e. routes aren't adapted depending on traffic conditions, which makes the approach scalable and stable. Congestion dependent decisions are made by the transport protocol.

So this “loose coupling” also achieves end-to-end traffic engineering at an RTT timescale, which is more dynamic than current TE and load balancing mechanisms like MPLS and ECMP. Further, a single flow isn't restricted to the maximum bandwidth of only one path. It may also mean that it's less critical for routing providers to select the best path, since several paths are exposed to the host.

So far we have considered resource pooling from the perspective of the end users (“maximise the social welfare of all users”). Considering it from the network perspective brings us on to accountability. The first aspect is the role of ECN marking. ECN marking is under the control of the network operator – it can decide when traffic on a link is such that it will start marking packets. So for instance a network operator selling a service class as “never congested” could use PCN (Pre-Congestion Notification), which marks packets as soon as the traffic rate climbs above (say) 75% of the available capacity. The MPTCP algorithms at the end points react to these congestion marks just the same as to any other congestion marks, and so the network operator could ensure that traffic was shifted out of this particular service class before it became actually congested. As a further example, imagine there are two networks, one “good” and the other “bad” (with low capacity). In a TCP world the “bad” network suffers lots of congestion, and the “good” network little. However, with MPTCP traffic naturally shifts into the “good” network. The “good” network can protect itself with PCN marking. A similar example is that a multihomed receiver could declare marks have been received over an interface that it wants to keep alive as a back-up but doesn't want to receive data on (perhaps it's more expensive).

Secondly, re-feedback enables “congestion transparency”, i.e. any node in the network can see the rest-of-path congestion (between this node and the receiver). This can help the operator decide where additional resources might be required for instance triggering some extra optical wavelengths to be lit or triggering connectivity to a new network. Also, “congestion transparency” enables an operator to get visibility of the end-to-end performance – just as end users do today. No longer does it have to worry that its efforts to improve its network may be ruined by a poor network somewhere downstream. The operator could actually use the re-feedback information to help judge which were really good paths to make visible to end users, rather than using unhelpful metrics such as AS path length as a heuristic indicator of good performance.

Thirdly, the accountability framework ensures “congestion truthfulness”. So end hosts really do have to react to the congestion marks – they can't just ignore them and do what they like – otherwise their packets get dropped (see Section 3.2). This should ensure that overall we really do solve the social welfare optimisation problem, and don't degenerate into the chaos of everyone grabbing as much of the resource pool as they can. Hence the accountability framework links back to the MPTCP, by ensuring that the voluntary congestion control algorithm really is obeyed. It can act as a counterbalance to the possibility of selfish actions leading to sub-optimal traffic distributions. It provides sufficient information that network provider will be able to trust the traffic engineering provided by the end hosts, reducing the cost and complexity of the network.



**Figure 15: Transparency of the whole path**

Fourthly, the accountability framework enables “cost transparency”. Fundamentally this is about ensuring that end hosts and networks bear the cost of the congestion they cause, by linking together causing congestion with making a payment (in some sense of “payment”) -- “congestion transparency” and “cost transparency” are two sides of the same coin. For example, a user can choose to send more traffic than the TCP algorithm allows, but it must pay – in some sense of “pay” – for the extra congestion it causes. The most obvious approach would be for users to make a micro-payment for every packet they send that arrives at the destination with a congestion mark. However this has two massive problems: the signalling overhead for every micro-payment; and users in practice seem to hate this kind of approach. Much of our work has therefore been how to make the “payment” practical and user-friendly: each user gets a flat-rate congestion allowance per month (similar to the volume allowance typical of an ISP contract today) and is policed within this allowance in an intelligent, dynamic fashion. Similar arguments apply at the border between networks, where their interconnect agreement could include bulk counting of marked packets across the interconnection, and perhaps any significant difference in the two directions is settled once a month.

In principle an operator can use these congestion payments to pay for network upgrades to alleviate the congestion. In fact, in a free market the payments are equal to the marginal cost of the upgrade. Multipath routing and transport enable a greater choice of routes and networks, and hence tend to make the market more free. Operators can recover costs associated with specific routes meaning that they could be encouraged to advertise routes other than the least cost routes – for example a high bandwidth satellite link may be rarely used due to cost; if these costs can be recovered then the route can be made available for bandwidth hungry, latency insensitive traffic. This works because after an upgrade the newly bigger network gains in two ways: (1) more traffic means it gets more revenue from ordinary volume or connectivity charging; (2) MPTCP will re-balance the total traffic across the resource pool and the bigger network will tend to get a bigger fraction of the traffic, and so as traffic grows more of the congestion revenue will tend to flow to the bigger network. To return to our example with a “good” and “bad” network, the “good” network gets paid when MPTCP causes a traffic surge from the “bad” network.

## 6 API Perspective

In this section, we consider the APIs required by the Trilogy architecture to address the three requirements of building the congestion signal, monitoring the congestion volume metric and interacting with the multipath transport.

### 6.1 Congestion Signal APIs

The accountability framework relies on congestion signals that are built in two stages, both of which are communicated at the IP layer: firstly the upstream congestion is added as data travels along the path and secondly the expectation of end-to-end congestion is re-inserted by the source. Each node with a bandwidth-limited bottleneck needs an interface to incorporate the local level of congestion into the upstream congestion signal. In the simplest set-up, this requires the network node to mark packets in proportion to the level of congestion of the scarce resource, as done by the ECN protocol. If sections of the path are tunnelled in IP or layer 2, then specific steps need to be taken to maintain the integrity of the upstream congestion signal, as it may be required by functions such as congestion volume monitoring which are located at domain boundaries.

Briscoe has addressed this topic for ECN signalling in the context of IP tunnels in [Briscoe09a]. To summarise: at the ingress of the tunnel, the congestion field is left unchanged in the encapsulated packet and the congestion field is copied to the outer header; at the egress of the tunnel, the congestion field of the decapsulated packet is updated to the maximum of the two congestion signals (hence the egress ensures that any marks that accumulate across the tunnel are reflected in the IP packet it forwards). If the end-to-end flow didn't support congestion signalling and the packet bears a congestion mark when exiting the tunnel, it should be dropped. The same approach should be used to deal with the re-ECN signal, so that congestion volume monitoring can still be performed at domain boundaries (which e.g. an IPsec tunnel may cross).

Operations are also required at end-systems. The destination needs to be able to feedback the congestion signal to the source whilst the source needs to know what the expected level of congestion is to re-insert into the forward path (note that this step is completely independent of the actual rate control algorithm that the source may be using which will also use the congestion feedback information to make decisions about how fast it can safely go). Finally, the congestion information also feeds into any congestion volume monitor on the path of the packet.

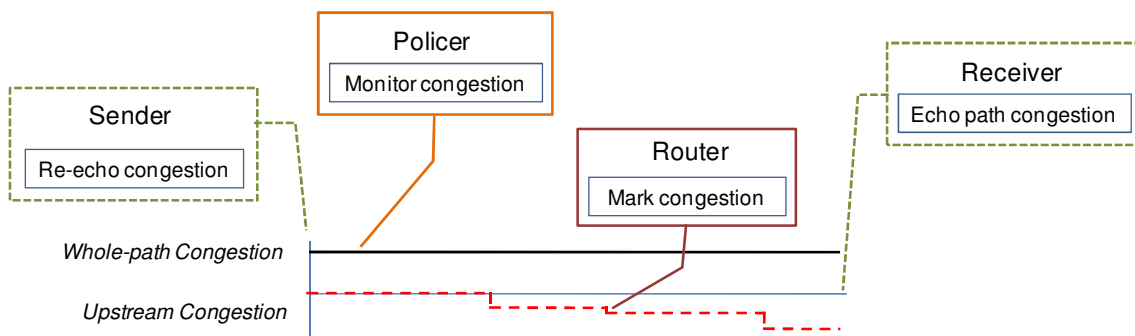


Figure 16: Interfaces to the congestion signal

### 6.2 Congestion accountability APIs

The congestion accountability framework (Section 3.2) has been designed around the contractual relationships between the different stakeholders involved in communicating over the Internet. The key element of the framework is the policer, which is operated at the attachment point between a customer

and its network provider. There is also a dropper element but this simply acts as a means to enforce honesty in the declarations of expected congestion levels.

### 6.2.1 Minimal set-up

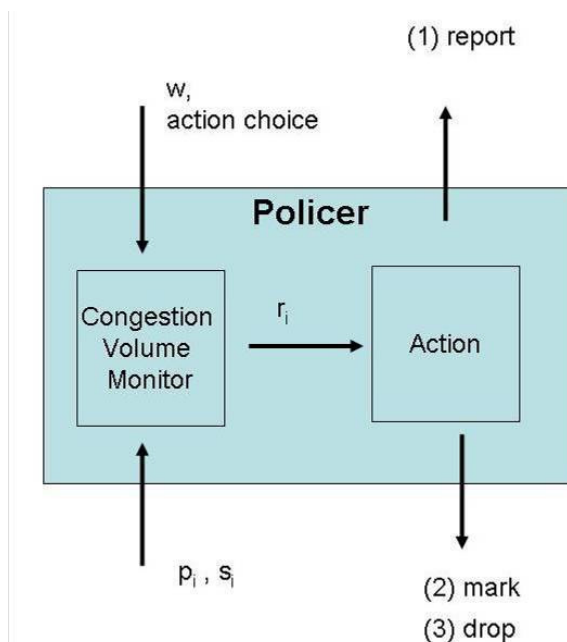
In the minimal set-up, congestion accountability requires policing at the attachment point between the customer and the network provider, based on the contractual agreement between the two stakeholders.

There are two interfaces to the bulk policer: the policy interface and the packet interface. The policy interface allows one to specify the congestion allowance given to the customer ( $w$ ) and the choice of any policing action. This API covers the parameters defined in the (contractual) agreement between the customer and the network provider; the essential parameter in all cases is the congestion allowance (token bucket fill rate). Secondary parameters include the congestion allowance roll-over (bucket depth), the depth of any overdraft, and the initial value of the token level.

A policy interface has to be defined for each contract between network provider and customer. In the particular case where a multi-homed customer obtains its network connectivity from several providers, its traffic would be divided between independent policers.

The packet interface with the network layer is limited to information available in the IP header:

- the size of the packet ( $s_i$ )
- the congestion signal ( $p_i$ )



**Figure 17: Simple congestion accountability API**

### 6.2.2 More advanced set-ups

If they wish to, customers can choose to set up their own “shadow” policer to better control their network experience. Interfaces to that policer are similar to the interfaces of the bulk policer. The main distinction is that the shadow policer requires the definition of rules to categorise each packet according to its priority. This requires one change to each of the interfaces:

- suitable configuration mechanism on the policy interface,
- extraction of IP header fields relevant to the categorisation of the packet ( $id_i$ )

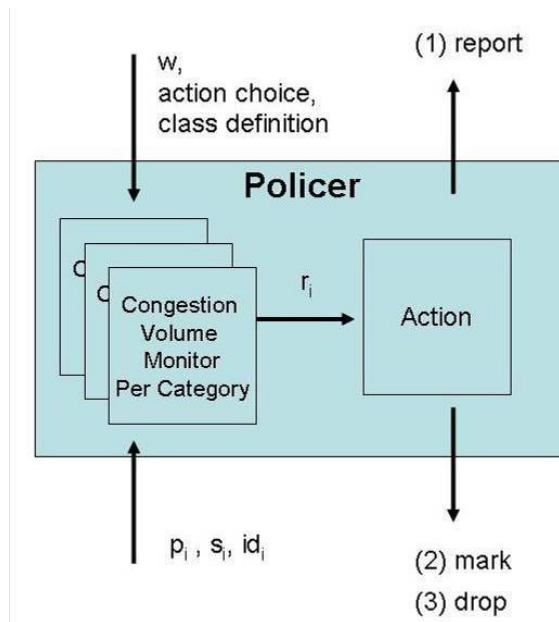


Figure 18: More advanced congestion accountability API

### 6.3 Multipath TCP API

The Multipath TCP (MPTCP) proposals as outlined in Section 3.1 are designed to be used in a basic form without the need for applications to be aware of its existence at all. It is envisaged that MPTCP would be enabled as a system-wide setting on a host, and any TCP sessions started by an application automatically attempt a MPTCP handshake first. A MPTCP session is still a byte-oriented stream, and from the point of view of data delivery an application should be agnostic about the use of MPTCP.

Nevertheless, in order to enable full exploitation of MPTCP features, and thus greater benefits of the resource pooling principle, it is desirable that applications are able to influence the behaviour of MPTCP. It is expected that such application preferences will be signalled using existing API calls, most likely `setsockopt`.

On a multihomed, MPTCP-capable host, this may take the form of an expression of preferences for MPTCP usage, e.g. “maximise throughput” or “minimise latency”. It is likely, however, that end users (via system preferences) should be able to override this if necessary. It would also be useful for MPTCP-aware applications to be aware they were using MPTCP and receive notifications about address changes, so that applications would be able to know the endpoint’s other address if it wished to initiate another connection, and also be informed about link property changes (e.g. for changing bandwidth usage on streaming media).

Future developments that are envisaged for MPTCP include partial or proxy MPTCP, where a multihomed node within a network implements MPTCP over a part of the network where disjoint paths are available. Such a proxy node may be as nearby as the host’s local router, or could be a significant way into the network. The end host may wish to express some preferences over link usage to this node, e.g. through prioritisation of traffic with DiffServ markings. Whilst such markings could be utilised to indicate which path to take, it is expected that the multihomed node will have a better understanding of the paths available to it and thus should do the scheduling, leaving the end host to use such markings for policy indication.



---

### 6.3.1 TCP semantics

We have begun to look into the semantics of MPTCP as compared to TCP. This is an important area to understand, since we need to retain backwards compatibility with existing applications.

Traditionally, a TCP connection is uniquely identified by a socket-pair (a host address and a port value). In MPTCP, a multihomed host will have several addresses that could be presented to the application. By default the first socket pair is presented to the application (i.e. the one associated with the first subflow); this is maintained even if the addresses change in the meantime. This may break applications that use TCP source and destination addresses to create other connections to the same host: if the initial address is “lost” (e.g. because of mobility), using this address to open new connections would not work as expected since the address is no longer available.

Alternatively, if MPTCP reported only one of the working socket-pairs, the source-destination addresses would change over time, which could break applications that use them for storing connection related values (e.g. using the hash of the source and destination sockets). The first option seems better. Over time, MPTCP-aware applications will be developed that, if they require detailed knowledge of addresses, will be able to query for this information with the extended API.

Below the TCP layer, IPsec can be used to encrypt connections between two hosts. If one TCP subflow is encrypted, and the application tries to open another subflow which may be in cleartext, that would affect the security gained with IPsec. If this is an issue then it can be solved by adding constraints on opening new subflows – either all subflows are protected by IPsec, or none.

At a high level, any middlebox on the path of a single MPTCP subflow cannot have access to the whole application data. This means it is impossible to properly reconstruct the data-stream – which may mean that existing network level intrusion detection systems will fail to catch attacks that are spread across multiple flows. MPTCP aware middleboxes would need to see all the subflows (thus be really close to the source or destination) to be able to properly reconstruct the data flow.

Load balancers are another type of middlebox that is used to spread request to multiple equivalent servers. With multipath TCP, subflows of the same client may be directed to different servers. If a subflow connection attempt has an unknown token, the server will reject it, and thus this situation is no worse than regular TCP, it just wastes SYN packets.

It is important to enumerate changes in the semantics of TCP header flags:

- **FIN:** The FIN flag only applies to the subflow on which it is used; a connection-level FIN exists to provide the semantics of a standard, data-level FIN.
- **ACK:** Acknowledgements are based on the subflow only, and the sender must map these acknowledgements to the data. This leads to slight changes in behaviour especially with session termination, discussed more in [Ford09a].
- **RST:** The RST (reset) only applies to a subflow. There is no connection-level RST, since it would be impossible to distinguish the two. This is because a RST is an indication that the sender has no knowledge of the TCP session that the received packet purports to be part of. If a host has no knowledge of a subflow (e.g. if the destination address has been reassigned), it also has no knowledge of the connection, yet the connection may still have working subflows on a different host (with different addresses), but the RST-sending host has no way of knowing this. A connection is considered reset if every subflow sends a RST in response.

### 6.3.2 Implementation of host policy

Of the set of paths/addresses available to a user, some may be more expensive than others – for instance 3G connections versus WiFi. In such situations users may want to limit the use of the expensive connections as long as cheap alternatives exist. This could be handled by system-wide

---

settings in the host that give preference to one or a few of the paths (this could be used for traditional hosts too).

An even simpler approach is that taken by current mobile phones (e.g. ones based on Symbian), which request permission from the user when an application wishes to access a network connection. Hence, the user can just stop the MPTCP using the more expensive connections, or indicate which interfaces they would like the MPTCP implementation to use.

The ability to make effective choices at the sender requires full knowledge of the path characteristics, which is unlikely to be the case, at least without re-ECN. Also receivers will often be the multihomed party, such as in the case of laptop computers with wired and wireless connectivity, so ideally the receiver should have the ability to signal its own particular preferences for paths. Instead of incorporating complex signalling, it is proposed to use existing TCP features to signal priority implicitly. If a receiver wishes to keep a path active as a backup, but wishes to prevent data being sent on that path, this could be achieved by the receiver not sending ACKs for any data it receives on that path. The sender would interpret this as severe congestion or a broken path and stop using it. We do not advocate this method, however, since this is brutal and will result in unnecessary retransmissions.

One proposal is to use ECN [Ramakrishnan01] to provide fake congestion signals on paths that a receiver wishes to stop being used for data. This has the benefit of causing the sender to back off without the need to retransmit data unnecessarily, as in the case of a lost ACK. This should be sufficient to allow a receiver to express their policy, although does not permit a rapid increase in throughput when switching to such a path. However it should be noted that this may corrupt the congestion accountability on this path and so great care should be exercised here. A potential solution to this would be that, if there is significant congestion, or the set of available paths has changed, MPTCP should wipe all subflow state and restart the multiplicative increase on all paths that appear uncongested. ECN will stop any paths that are still not required immediately, while the receiver's desired backup path will be in use and throughput will increase quickly. This proposal should be no worse than current TCP.



## 7 Conclusions

The Internet architecture has lasted forty years and few real changes have been deployed in that time, despite many years of research. The difficulty of changing the Internet cannot be underestimated, so our goal to “re-architect the Internet” is highly ambitious. We believe that we have made good progress with steps towards the Future Internet, with four Design Principles for architectural developments, three complementary architectural building blocks, and an overall framework within which they can complement each other.

At the start of this work we set ourselves two overarching goals for the Trilogy architecture to fulfil:

- It should be ‘designed for tussle’. It can adapt in a scalable, dynamic, autonomous and robust manner to local operational and business requirements. It can avoid prejudging the commercial and social outcomes for the different stakeholders.
- It should be the hourglass for control, with a ‘narrow waist’. Over recent years the Internet has accumulated many control mechanisms, to cope with new requirements on the Internet, as it shifted from best effort communications between trusting, static hosts to resilient, time-sensitive communications between mobile, non-trustworthy hosts and servers. New control mechanisms also enable new business models – so linking back to ‘design for tussle’.

**Design for Tussle** – Our design process has tried to ensure our architecture is tussle-friendly. We began by looking at the broad socio-economic goals for the future Internet – things like flexibility, accountability and sustainability – that go beyond the original Internet design principles such as the ‘end to end principle’ [Clark88]. We translated these goals into a set of practical and useful **Design Principles**: concrete guidelines usable by protocol designers, network architects and Internet engineers to help them achieve a solution that is ‘designed for tussle’:

- *Information Exposure Principle*, the principle that sufficient information about resource usage should be exposed to support an effective and efficient allocation, in a timely fashion. To achieve this, the information is integrated into the data (or transaction) that uses up the scarce networking resource.
- *Separation of Policy and Mechanism Principle*, the principle that higher level policy decisions – the implementation freedom – are separate from the standardised mechanisms for implementing control through the exchange of information.
- *Fuzzy Ends Principle*, the principle that endpoints are able to explicitly delegate some functions to the network.
- *Resource Pooling Principle*, the principle that resources in the network should be able to be pooled, so that they can effectively be utilised as a single resource, in order to improve the effectiveness and efficiency of the network.

Turning our attention to Trilogy’s technical scope (reachability and resource control at the network and transport layers of the standard protocol stack), we are developing three **architectural ‘building blocks’** cognisant of our design principles:

- *Multipath TCP*, an end-to-end transport protocol that can exploit multiple paths, for instance to take advantage of multihoming at endpoints in order to improve reliability and utilisation in the network. As TCP is currently the most widely used transport protocol, we have chosen to adapt TCP to become multipath capable. We are working on two protocol designs, with complementary use cases: two-ended, where both endpoints are aware of the multipath TCP and so can negotiate any combination of available paths; and a one-ended design, which is targeted at large sites that are multihomed but single-addressed, allowing rapid realisation of resource pooling benefits with only one end requiring upgrading.

- *Multipath routing*, protocols that allow routers to select multiple routes to a given destination, so as to improve reliability and resource pooling inside the network. We are working on extensions to both existing intra-domain routing protocols and to BGP.
- An *accountability framework* for resource usage that ensures end-users and network operators can be held to account for the impact their actions have on everyone else that uses the network. We are working on the re-feedback protocol that achieves ‘congestion transparency’ – which means that all nodes can see - and declare truthfully - the congestion they cause by forwarding (or sending) traffic.

To summarise how the three architectural ‘building blocks’ reflect the Design Principles. Firstly, multipath TCP and multipath routing both enhance *resource pooling*. Resource pooling gives the network improved resilience and adaptability, hence allowing a wider range of outcomes where the network can exist. In particular it allows the possibility of greater competition (which [Clark05] considers part of design for tussle); as a specific example, multipath TCP enables an end user to reach multiple access operators through one interface – a kind of virtual local loop unbundling. Secondly, the accountability framework makes the network ‘cost transparent’ – in terms of *information exposure* it allows a particular user and/or network to see the underlying ‘cost’ and decide whether to ‘pay’ or adapt in some other way. For example a user might prioritise its important applications when ‘costs’ are high, or an operator might use DPI to do the prioritisation on behalf of the end user (perhaps in corporate environments, to control the employee’s experience, or in mobile networks, to penalise VoIP say).

**Hourglass for control** – We have deliberately not tried to devise a Grand Unified Architecture that covers all potential control functions, present and future – an impossible task. Instead we have worked on individual architectural building blocks, which can be “loosely coupled” - so allowing independent deployment whilst complementing each other and bringing cumulative benefits. We have considered specific interactions between the three building blocks and how they can complement each other.

For each building block we have tried to “design small but think big”. “Design small”, in that the new function should be lightweight and simple (so adding little “fat” to the control waist), it should not break what is already there and should be ease to deploy. “Think big”, in that the new function should have a significant impact towards alleviating the severe pressures on the Internet from new applications, new business models and new networking technologies. In this context, we review each of our building blocks in the following three paragraphs.

Multipath TCP is designed so that as far as applications are concerned it’s just like TCP (the API is essentially unchanged) but with better performance. If only one of the end hosts is MPTCP-enabled, then either it falls back to ordinary TCP or else one-ended multipath TCP is used. Several design features have been determined by deployability (e.g. NAT considerations), and these are now under active debate in the broader IETF community. The impact of multipath TCP should be far-reaching, making major resilience and utilisation benefits open to all applications and users.

Multipath routing is designed as extensions to existing protocols: again for ease of deployability, we have chosen not to work on a new routing protocol, rather to tweak existing widely-used protocols such as BGP. The impact of multipath routing is better resilience and utilisation via resource pooling.

The accountability framework is designed to require the minimum of changes to IP. It re-uses ECN marking and only needs one bit in the IP header. We recognise that getting even this one bit assigned is very challenging, so we are currently working out an interim step that allows wide-scale experiments on the public Internet. Although the accountability framework also needs a policer box at the network’s edge, the deployment incentives are aligned – the operator is protecting their network. The impact of the accountability framework is large: networks have the incentive to invest in more network capacity, knowing that they have simple yet powerful transparency about who is using it; and users can adapt according to the ‘value’ of a particular flow and the ‘cost’ of the current congestion, so bringing true end-to-end QoS.



---

We believe that deployment of our three architectural building blocks – whether singly or in combination - should thin down the existing ‘fat waist’ of the hourglass for control, by enabling some “liposuction” on functions like QoS, TE, DPI. But there are plenty of control functions outside the Trilogy project’s scope, such as mobility and security, where we hope others can make progress by following our Design Principles and general approach.

## **7.1 Future work**

Our work now concentrates on refining and validating our three proposed architectural building blocks. We are developing detailed technical protocols. Validation work includes a combination of discussion with other Internet experts, especially at the IETF, engineering evaluation, mathematical analysis, prototyping and simulation.

There are many items worthy of future study. Amongst these are: a single path transport protocol that can nevertheless utilise a multipath network by changing the path when necessary; a weighted congestion control algorithm that can adapt according to the priority of the data; a deployment path for the accountability framework; a stability analysis of the various proposals for multipath routing; and an analysis from a socio-economic perspective of the various technical proposals. Open questions include: how the multipath transport protocol can see and influence the multipath routing choice; how diverse multiple paths could be on the Internet; and whether congestion volume is a sufficient metric.

## **7.2 Concluding Remarks**

This document set out to define the overall architecture adopted by the Trilogy project as a step towards the over-arching goal of re-architecting the Internet: our long-term objective is to change the future Internet. We are developing detailed technical proposals in three areas: multipath TCP, multipath routing and an accountability framework. We have presented our architectural approach, namely our Design Principles and our holistic approach towards network and transport control, which we hope others will adopt. Our ambition is to develop an architecture for change - an architecture that can adapt in a scalable, dynamic, autonomous and robust manner to local operational and business requirements - an architecture that avoids prejudging the commercial and social outcomes so that it can reflect the outcome of contentions amongst the Internet’s stakeholders. When more fully realised, we believe our results will significantly enhance the reliability, robustness, manageability and functionality of the Internet, and will create new and varied business opportunities based around a common control architecture.

---

## References

- [Acemoglu07] D. Acemoglu, R. Johari, and A. Ozdaglar, “Partially optimal routing. IEEE Journal of selected areas in communications”, 2007
- [ALTO08] “Application-Layer Traffic Optimisation (ALTO) BoF”, IETF 72, Dublin, July 2008, <http://www3.ietf.org/proceedings/08jul/agenda/alto.html>
- [Bagnulo09] Joint Multi-path Routing and Accountable Congestion Control, Marcelo Bagnulo, Louise Burness, Philip Eardley, Alberto García-Martínez, Francisco Valera, Rolf Winter, ICT Mobile Summit 2009, Santander, 10 - 12 June 2009
- [Beijnum09a] I. van Beijnum, J. Crowcroft, F. Valera and M. Bagnulo. Loop-freeness in multipath BGP through propagating the longest path. International Workshop on the Network of the Future (Fut-Net 2009). June 2009, Dresden, Germany
- [Beijnum09b] I. van Beijnum, One-ended multipath TCP, IETF Internet-Draft <draft-van-beijnum-1e-mp-tcp-00.txt> (May 2009).
- [Braden98] B. Braden et al., Recommendations in Queue Management and Congestion Avoidance in the Internet, RFC2309, April 1998
- [Briscoe05] B. Briscoe, A. Jacquet, C. Di Cairano-Gilfedder, A. Salvatori, A. Soppera and M. Koyabe, “Policing Congestion Response in an Internetwork using Re-feedback”, Proc. of ACM SIGCOMM, Philadelphia, PA, USA, Sep 2005
- [Briscoe07] B. Briscoe, “Flow rate fairness: Dismantling a religion. ACM SIGCOMM Computer Communications Review, 37(2): 63-74, April 2007
- [Briscoe08a] B. Briscoe, “Byte and Packet Congestion Notification”, Internet Draft, August 2008
- [Briscoe09a] B. Briscoe, Tunnelling of Explicit Congestion Notification, IETF Internet-Draft <draft-ietf-tsvwg-ecn-tunnel-02.txt> (Mar 2009).
- [Briscoe09b] B. Briscoe, A. Jacquet, T. Moncaster, A. Smith. Re-ECN: The Motivation for Adding Congestion Accountability to TCP/IP. draft-briscoe-tsvwg-re-ecn-tcp-motivation-00. March 2009
- [Briscoe09c] B. Briscoe, A. Jacquet, T. Moncaster, A. Smith Re-ECN: Adding Accountability for Causing Congestion to TCP/IP draft-briscoe-tsvwg-re-ecn-tcp-07 March 2009
- [Burness09] The Trilogy Architecture for the Future Internet, Louise Burness, Philip Eardley, Robert Hancock. In the book “Towards the Future Internet - A European Research Perspective”, ISBN 978-1-60750-007-0, May09
- [Carpenter96] B. Carpenter (ed.), “Architectural Principles of the Internet”, RFC 1958, June 1996
- [Clark05] D. Clark, J. Wroclawski, K. Sollins, R. Braden, “Tussle in Cyberspace: Defining Tomorrow’s Internet”, IEEE/ACM Transactions on Networking, 13(3), p. 462-475, June 2005.
- [Clark88] D. Clark, “The design philosophy of the Darpa Internet protocols,” In Proc. ACM SIGCOMM, Vancouver, BC, Canada, Sept. 1988.



- 
- [Dukkipati05] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, N. McKeown, “Processor Sharing Flows in the Internet”, IWQoS 2005
- [Eardley09] P. Eardley (ed.), “Pre-Congestion Notification Architecture”, RFC5559, May 2009
- [Ford09a] Ford, A., Raiciu, C., Handley, M. and Barré, S., “TCP Extensions for Multipath Operation with Multiple Addresses”, draft-ford-mptcp-multiaddressed, May 2009
- [Ford09b] Alan Ford, Philip Eardley, Barbara van Schewick, New Design Principles for the Internet, IEEE ICC Future networks 2009, June 14-18, Dresden, Germany
- [Friedman98] E. Friedman, P. Resnick, The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy*. 10(2) 173-199, 1998
- [Gao00] L. Gao and J. Rexford Stable Internet Routing Without Global Coordination, in *Proc. ACM SIGMETRICS Conf (2000)* 681-692
- [Gibbens99] R. J. Gibbens, F. P. Kelly, “Resource Pricing and the Evolution of Congestion Control”, *Automatica* 35, 1999
- [Handley08] M. Handley, D. Wischik, M. Bagnulo, The Resource Pooling Principle, *ACM CCR (Computer Communication Review)*, Oct 08.
- [Jacquet08] Policing Freedom - to use the Internet Resource Pool. Arnaud Jacquet, Bob Briscoe & Toby Moncaster. *ReArch'08 - Re-Architecting the Internet*, December 9th, 2008, Madrid
- [Katti06] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, J. Crowcroft, “XORs in the air: practical wireless network coding”, *ACM SIGCOMM Computer Communication Review*, Volume 36 , Issue 4, October 2006
- [Kelly98] F. P. Kelly, A. Maulloo, D. Tan, “Rate control in communication networks: shadow prices, proportional fairness and stability”, *Journal of the Operational Research Society*, 1998
- [Kohler06] E. Kohler, M. Handley, S. Floyd, “Designing DCCP: Congestion Control Without Reliability”, *SigComm* 2006
- [Kukec09] A. Kukec, M. Bagnulo, M. Mikuc, SEND-based Source Address Validation for IPv6 10th International Conference on Telecommunications (ConTEL) Zagreb, Croatia, June 2009
- [Laws92] C.N. Laws, “Resource pooling in queueing networks with dynamic routing”, *Advances in Applied Probability* 24 (1992) 699-726.
- [Mérindol07] P. Mérindol, J-J. Pansiot, S. Cateloin. Path Computation for Incoming Interface Multipath Routing (2007), *IEEE ECUMN*.
- [Mérindol09a] P. Mérindol, J-J. Pansiot, S. Cateloin. Low Complexity Link State Multipath Routing (2009), in *Proc. of Global Internet Symposium*
- [Mérindol09b] P. Merindol, J-J. Pansiot, S. Cateloin. The mDT algorithm (2009), [arXiv:0904.0217v1](http://arxiv.org/abs/0904.0217v1), LSIIT-UdS Techreport
- [Mitzenmacher01] M. Mitzenmacher, “The Power of Two Choices in Randomised Load Balancing”, *IEEE Transactions on Parallel and Distributed Systems*, Volume 12 , Issue 10 (October 2001)
- [Mrinfodata] J.-J. Pansiot, The mrinfo project, <http://svnet.u-strasbg.fr/mrinfo/>.

- 
- [Nordmark09] E. Nordmark, M. Bagnulo, First-Come First-Serve Source-Address Validation Implementation, Internet-Draft, draft-ietf-savi-fcfs-01, March 2009.
- [P4P] The P4P Working Group, <http://www.openp4p.net/front/p4pwg>
- [Ramakrishnan01] K. Ramakrishnan, S. Floyd, D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. September 2001. RFC3168
- [Rekhter06] Y. Rekhter, T. Li, S. Hares, A Border Gateway Protocol 4 (BGP-4), RFC 4271. January 2006
- [Roughgarden02] T. Roughgarden and E. Tardos, “How bad is selfish routing?”, Journal of the ACM, 2002
- [Saltzer84] J. Saltzer, D. Reed, and D. Clark, “End-to-end arguments in system design”, ACM Transactions on Computer Systems, 2(4):277-288, Nov 1984.
- [Trilogy08-D2] “Lessons in ‘Designing for Tussle’ from Case Studies”, Trilogy Deliverable D2, May 2008
- [Trilogy08-D3] “Initial Overall Architecture”, Trilogy Deliverable D3, August 2008
- [Trilogy08-D4] “Design space analysis for reachability”, Trilogy Deliverable D4, December 2008
- [Trilogy08-D5] “Design space analysis for resource control”, Trilogy Deliverable D5, December 2008
- [vanSchewick09] B. van Schewick, “Architecture and Innovation: The Role of the End-to-End Arguments in the Original Internet”, forthcoming 2009, MIT Press
- [Vutukury99] S. Vutukury and J.J. Garcia-Luna-Aceves A Simple Approximation to Minimum-Delay Routing, in Proc. ACM SIGCOMM Conf (1999) 227-238
- [Walton09] D. Walton, A. Retana, E. Chen and J. Scudder, Advertisement of Multiple Paths in BGP, draft-walton-bgp-add-paths-06.txt, IETF Draft, 2009



---

## Annex A Design principles

This Annex presents four engineering-level Design Principles intended to help meet the requirements of Design for Tussle, and is largely drawn from [Trilogy08-D3]. The design principles are also discussed in [Ford09b].

### A.1 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/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 the ‘control information’. 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.

Of course usage of resources which aren’t scarce will (probably) need to be accounted for, but this information doesn’t need to be “integrated” (see below) with the data /transaction. Also, resources which are purely used and optimised within a single system aren’t relevant<sup>7</sup>, e.g. resources purely at the end system (for the data /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 the appropriate timescale)
- In space (over the appropriate participants)
- In function (for the appropriate activity)

Obviously integrating the control information leads to an additional cost in transmitting that information and careful thought needs to be given to ensuring the benefits of the information outweigh its costs. This may for instance pose a limit on the precision of the information.

**Reflects its resource usage in real time** emphasises the first of these points, i.e. the granularity of the information in time should reflect that over which the data /transaction depletes the resources (in other words, the timescale it imposes load for). 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 it is the lifetime of the binding state.

For some types of resource, it is simply the 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 (most routers are ‘byte congestible’).

---

<sup>7</sup> 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.

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 the Design Principle gives the reason for the control information ('support an efficient and accountable allocation of resources') and thus guides how much detail the information needs to have.

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. So we should consider carefully where the extreme case of "perfect competition" would drive us. The actual pricing formula may be off-line and may be sophisticated.

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.

For a transaction, it may not be immediately obvious how much resource the transaction will consume, e.g. how much processing will it take to process the message? At one level, uncertainties are risks, which can also be handled as a cost. One solution to this is to look at the information as an expectation of the costs likely to be incurred. You can then use the accountability framework to impose penalties on those users who are overly optimistic in their predictions of usage.

## A.2 Separation of Policy and Mechanism

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

**Allow a network entity local choice** It is important to ensure there is freedom in any implementation – inflexible designs seldom have any longevity. The 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 and in light of changing circumstances). Implicitly, policies are independent of each other. On the other hand the mechanism does need to be standardised.

**Have a common protocol or mechanism** A good example is Kelly's view of congestion control [Gibbens99]. The mechanism is congestion pricing, which represent the externality cost that a packet as on other users. It is the end user's 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). It is also 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 [Dukkipati05]) would break this Design Principle. So too would an approach where the user requests a specific rate which is then arbitrated by the network.

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 [Eardley09]. Another example is DCCP [Kohler06]. It defines a three way handshake for connection set-up (mechanism), which allows the ends to negotiate which congestion control algorithm (TCP, TCP friendly etc) to use for the data transfer phase. 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 can’t ruin the service received by other users. We believe that enough constraint is set by the Information Exposure Principle, i.e. the mechanism needs to integrate the control information (e.g. metric) that reflects the usage of scarce resource by the mechanism’s data (or transaction). For example, policy about what granularity prefixes should be advertised/accepted (e.g. /16s or /32s?) – too fine a granularity causes too high a cost in terms of volume of messages and processing. In other words the policies are constrained to be sane by accountability.

A key point to note is that it may be necessary to separate different mechanisms into different layers since they may lead to conflicting answers. 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.

### A.3 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”.*

First, some examples of the sort of delegation we’re referring to; these are things that all could be done by the endpoint, 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). Another example could be to 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, as suggested by [ALTO08], [P4P].

The “endpoint” might, for client server communications, be either end.

**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<sup>8</sup>. For trust and accountability reasons it seems most likely that the delegation will be over a single administrative hop.

**Endpoint** and **network** denote a functional relationship between computers that ‘use’ the network and computers that ‘provide’ the service offered by the network (i.e. data forwarding). So endpoints may well be administered by a network provider or be topologically in the network.

One issue for the Design Principle is the constraints there are 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

<sup>8</sup> Claims about equity or the usefulness of the service can be dealt with by competition and/or regulation.

and implement their functionality at layers above the Internet layer” [vanSchewick09]. 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.

The 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 and 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 and make sure that the system can evolve by drawing these functions back into the endpoints. The “distributed system” view attempts to achieve the above points.

Our fuzzy ends Principle is, unsurprisingly, closely related to the original Internet end to end Design Principle. As [vanSchewick09] points out, the end to end principle in fact comes in two versions. The “narrow” version of 1984, 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. The “broad” version of 1998, which says lower layers (the middle) should only provide very general services that can be used by all applications; optimising support for one application hinders long-term system evolvability, application autonomy and reliability. The view that this Design Principle is restricted to application-level intermediaries is compliant with the “broad” version. The alternative view isn’t restricted like this. In terms of the “narrow” version of the end to end principle it argues that functions can be implemented in the middle also for business /social reasons; and it modifies the “broad” version in a confined, controlled manner so as to minimise loss of its benefits.

## A.4 Resource Pooling

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

**Pooling of resources** means making the network’s resources behave like a single virtual 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 an analogue 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) e.g. [Laws92]
- Google (and other CDNs) pools together the processing cycles, bandwidth and reliability of all its distributed servers
- A wireless system like CDMA pools the bandwidth, power and interference of its base stations and mobile terminals
- Network coding pools multiple messages into fewer messages over a pool of links [Katti06].
- 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 localised surges in traffic;
- maximised utilisation.

*To be effective* means there is enough resource pooling to bring about the benefits above. So there doesn't have to be unlimited relocatability. The "power of two choices" [Mitzenmacher01] 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, for example with ECMP a specific flow follows a deterministic path, but overall it achieves reasonable load balancing.

*Conflict* means that two resource pooling mechanisms may 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; it can be shown that the cost of anarchy (i.e. the degradation in performance due to conflicting load-shifting) can be arbitrarily high [Roughgarden02], [Acemoglu07]. A conflict might also be between two instances of the same resource pooling mechanism fighting to gain a bigger share of the pooled resource. It is assumed that conflicts can be handled via careful design (case by case analysis) and via the Information Exposure Principle. Note that such conflicts are not the same as the tussles that are a necessary part of Design for Tussle.