On Designing for Tussle: Future Internet in Retrospect

Costas Kalogiros[1], Alexandros Kostopoulos[1], and Alan Ford[2]

[1] Athens University of Economics and Business, Department of Informatics,
76 Patission Str., Athens 10434, Greece
{ckalog,alexkosto}@aueb.gr
[2] Roke Manor Research, Old Salisbury Lane, Romsey, SO51 0ZN, United Kingdom
alan.ford@roke.co.uk

Abstract. Over the past decades, the fundamental principles of the Internet architecture have not significantly changed. However, Internet evolution and its effects on participants' interests have triggered the need for re-defining these design principles. "Design for Tussle" is an aspiration for future network designs, which enables the involved stakeholders to express their possibly conflicting socio-economic preferences on service instances. We performed a series of case studies examining whether established technologies are compatible with this new approach. Using the knowledge gained, we provide canonical examples and help protocol and network designers better to consider how to come up to the problem of "designing for tussle" in order to realize a flexible architecture. Finally, we associate protocol success to adoption and show, using empirical evidences, that carefully embracing the "Design for Tussle" paradigm can outweigh the higher complexity in protocol design.

Keywords: Design for Tussle; Future Internet Architecture; Network Protocols; Technology Adoption; Case Studies.

1 Introduction

The Internet today is a playground of many competing forces (technical, economical and social), where different stakeholders with possibly conflicting interests interact with each other. These ongoing "tussles" may constitute a threat to the architectural integrity of the Internet. Researchers, service providers, network operators and users have recognized that the current Internet architecture is ill-suited to satisfy the demands and requirements of our modern society [8]. The fundamental design principles of the Internet architecture, designed decades ago, are currently under increasing evaluation [3].

It is suggested that the future Internet architecture should incorporate the necessary flexibility to adapt to changing economic and social stresses, the so-called "Design for Tussle" principle. This new paradigm recognizes the necessity for traditional design goals – such as protocol correctness – to be satisfied, but proposes that socio-economic ones should also be considered. Clark et al. [5] proposed an initial set of design principles that can be used to accommodate tussles, these being to "Modularize along tussle boundaries" and "Design for choice".

M. Oliver and S. Sallent (Eds.): EUNICE 2009, LNCS 5733, pp. 98–107, 2009.

Meeting these two, more specific, design principles leads to a system that is able to flex under pressure and survive, even if stakeholders and the environment constantly changes. The ultimate goal of these design principles is to allow for "variation in outcome", instead of promoting a unique solution that may not be aligned with all legitimate participants' opinions. For example, protocols that are "designed for tussle" support many business models instead of a single one that the designer found to be attractive. In this way, the outcome can be determined by the interaction of all stakeholders. Of course, all legitimate participants should have the freedom to express their preferences. As an example, a provider could choose to offer a "walled garden" service if she finds it valuable. But, the designer should not bias the outcome, even if all evidence shows that this leads to a socially optimum equilibrium. History of the Internet, so far, has shown that we cannot predict the consequences when we build protocols based on assumptions for the future.

Furthermore, such an approach would set the stage for the Internet to operate more freely, without the need for regulatory intervention to battle anti-competitive tactics from powerful participants. This competitive setting is achievable if all stakeholders have the potential to exercise some sort of control, using the same or complementary protocols (for example select their provider from a list of candidates).

While Clark's paper provides the foundations for a tussle-aware architecture, it is far from obvious how such tussles can be incorporated into the Internet and how all derived principles can be applied to an architectural design. Besides, the task of protocol design in such all-encompassing platforms is already extremely complex, requiring special skills and systematic approach. Many believe that designing system components is an art rather than a science. We suggest that one should carefully balance the trade-off between traditional protocol design goals (i.e. performance) and socio-economic ones (i.e. flexibility).

We try to reduce this inherent difficulty of "designing for tussle" in two ways:

- First, we try to shed some light on the details of applying the two more specific design principles mentioned above. We do this by giving examples of functionality in established Internet protocols that, intentionally or not, meet or violate these design principles. We, also, try to give some guidance to designers by providing additional criteria that should be met.
- Furthermore, we try to justify the extra difficulty imposed on designers and standardizing organizations to embrace this new paradigm. We do this by trying to correlate the outcome over time of protocol adoption (or abandonment) to their "score" against these design principles.

In order to achieve our goal, we performed a systematic analysis of interesting case studies, from a broad commercial and strategic viewpoint. These protocols were carefully selected in order to cover functionality ranging from network to application layer. In particular, we investigated HTTP, BGP, TCP, NAT, IPv6, SIP and ENUM.

The paper is structured as follows: We give an overview of related work in Section 2. Sections 3 and 4 present a high level characterization of the above case studies with respect to the two specific design principles. In particular, Section 3 attempts to clarify how modularized protocols can be designed, and Section 4 discusses protocols which are designed for choice for example through the use of open interfaces. Section 5 correlates adoption issues of recent technology developments

and proposals to their compatibility with the "design for tussle" paradigm. Finally, we conclude our work in Section 6.

2 Related Work

Saltzer et al. [11] described the fundamental design goals underlying the current Internet and the resulting design principles. These original design goals and principles have led to the current hourglass architecture, where IP provides a common layer between the transport and higher layer protocols and the disparate lower-layer communications technologies. This approach has largely contributed to the successful operation and expansion of the Internet. In particular, the "end-to-end" principle [4] was one of the central design principles of the Internet.

Over recent years, researchers have increasingly argued that the design goals and principles must be critically reviewed to ensure that the Internet continues to operate [10]. Moreover, new design principles may be needed that were not thought of for the original design of the Internet. The most notable recent principle proposed is the "design for tussle" principle, raised by Clark [5]. Later, in [6] and [13], new principles were presented for future Internet architecture; the "information exposure", the "separation of policy and mechanism", the "fuzzy end-to-end" and the "resource pooling" principle. These principles have particular focus on enabling socio-economic tussles between stakeholders.

The term "tussle" is described as an "ongoing contention among parties with conflicting interests". The Internet is increasingly used as a space where conflicts of interests arise and the different players – including users, ISPs, service providers, governments, etc. – are battling over the control for economic, social or political reasons. That tussles are not necessarily negative. Instead, they are needed to allow evolution and progress. Architects and engineers should understand the rules that define the tussles in order to shape the architecture and to ensure evolvability. In [5], more specific principles for "design for tussle" are identified. "Modularization along tussle boundaries" aims to break down the complexity of the tussle, and suggests that functions within a "tussle space" (a "place" where conflicts of a specific kind of interests occur, i.e. security) should be logically separated from functions outside of that space. It is also identified that protocols should be "designed for choice" in such a way that all the parties to an interaction have the ability to express their preferences about which other parties they interact with.

3 Tussle Isolation

The goal of *isolation of tussle* aims to ensure a separation of tussle spaces, so that tussles can occur independently of each other. According to this design principle, the function that allows a tussle to be played out should have minimal impact on other tussles, and therefore also on stakeholders that are not directly related to this tussle. This is achieved through "modularization along tussle boundaries", which is fairly simple to define, but a hard task to implement.

A useful way to think about and support modularization is to distinguish between "functional" and "stakeholder" separation. "*Functional separation*" is the creation of tussle spaces bounded according to functions, which are logically separated from functions that lie outside of this space. "*Stakeholder separation*" is separation between stakeholders, within a functional tussle space, i.e. allowing players to act with minimal dependence and keep their internal choices separate from external stakeholders. This is often closely related to functional separation, depending upon where the boundaries of the tussle spaces are defined. The following examples illustrate varying degrees of success or failure in achieving this functional and stakeholder separation.

HTTP provides a good example of a clean, simple modular design, separating functions and allowing natural protocol evolution without affecting other functions. In particular, the separation of header and data body allows extensibility without affecting the data being delivered. Responsive web applications and object-oriented services, such as those driven by AJAX, PHP and SOAP, use HTTP to deliver dynamic content, without changing the protocol.

The inter- and intra-domain routing system is a clear example of separation based on stakeholders and functions at the same time. In particular, the split between intra- and inter-domain routing allows different protocols to be used in the interior, depending on a domain's needs, while maintaining a consistent exterior presentation (in the form of BGP messages). This allows interior routing protocols (such as RIP, OSPF, etc.) to evolve, or be completely replaced, with no effect on connectivity with the rest of the Internet. As a result, each domain acts independently of the others.

However, sometimes modular design is difficult to achieve, like the case of Network Address Translators. NATs were originally developed as an administrative aid, so that networks could manage their internal hosts and addressing independently of their providers. In particular, this greatly assists in renumbering either address space (including changing provider), or adding new hosts internally without any negotiations with the upstream provider. This initial modularization was a stakeholder separation, whereby external (provider) and internal (customer) address spaces were decoupled. At the same time, the growth of the Internet was leading to potential IPv4 address exhaustion, and so NATs began to be used to slow the rate of consumption of IPv4 addresses. However, the tussle over address allocation expanded into the trust space, because NATs also protect against malicious activity initiated by external hosts. Furthermore, NATs began to have many unintended consequences on other stakeholders. NATs break end-host reachability, and thus limit innovation by restricting nodes behind a NAT to use supported protocols only, and not to operate servers. Some applications (such as Skype) with no direct impact on the original tussles of address allocation are also adversely affected. Certain workarounds, such as NAT pinholes (a.k.a. "port forwarding"), have been used to reduce the impact of this; however end users are required to be proactive in working around these issues.

IPv6 also suffers from poor functional modularization. Although its original function was also to provide an expanded address space, many other features were included as standard (such as host auto configuration, and originally mobility and security features, although these are no longer mandatory), and as such the sheer weight of the "base protocol" module makes its deployment a very expensive task. A larger amount of functional separation could have eased these issues, improved

incremental deployment possibilities, and could have even facilitated backwards compatibility. For example, DHCPv6 could have been implemented as an entire modular replacement for the standard router discovery. Similarly, IPv6 suffers from poor stakeholder separation, since the use of IPv6 by one stakeholder is only of use if other stakeholders (endpoints, transit providers, software authors, etc.) also adopt it.

The design of TCP is modularized to some extent. TCP is one of the core protocols of the Internet, providing reliable end to end transmission of packets, and trying to avoid congestion occurring inside the network. Especially for the latter function, there are different implementations proposed (TCP Tahoe, Reno, Vegas, etc.) for the Additive-Increase-Multiplicative-Decrease (AIMD) scheme in order to control the transmission rate. This is functionally separated from, for example, the reliability features of TCP. These functions are, however, linked elsewhere, reducing the benefit of this separation. The occurrence of packet loss is an overloaded signal, as it is also used to detect congestion by existing TCP control mechanisms, despite the implementation of the algorithm being entirely separate. Explicit Congestion Notification (ECN) [9] and Re-feedback [2] are proposals to use the network information in the transport layer to improve congestion control, separately from the dropping of packets. In particular, Re-feedback proposes a change to the TCP/IP feedback architecture as an attempt to design for tussle for Internet congestion control. Both these mechanisms allow network elements to know the congestion on the downstream path, i.e. between the network element and the destination. Such mechanisms aim to separate congestion control from data transfer and error detection.

Finally, the design of SIP (Session Initiation Protocol) and Public ENUM (tElephone NUmbering Mapping) is modular to some degree, since they decompose the problem of calling a destination into two tasks: identifying a user, and calling the user. SIP is a signaling protocol for initiating and managing sessions such as VoIP calls, while ENUM helps the convergence of VoIP and circuit switching by providing mappings between different identifiers. This has successfully modularized these tasks, allowing alternative technologies to be used as the parties see fit (i.e. tussles to be played out), without altering the interface between the modules. In deployment terms, however, ENUM suffers from the same problem as given above for IPv6. It requires a number of stakeholders to enable it and expend time and effort configuring, deploying, and supporting it, in order for anybody to see a benefit. SIP, on the other hand, requires no additional technology beyond standard TCP/IP, and as such can be incrementally deployed by stakeholders with only limited cost before benefits can be realized.

4 Design for Choice

By modularizing the tussle boundaries we restrict the set of stakeholders that are affected by a protocol. The next step is to give each stakeholder the ability to influence the outcome of a tussle. This entails that each participant has the right to be given enough control during protocol's configuration and at "run time". Then, it should be her option whether to use this right in person, delegate it to a trusted third entity or disregard it completely. In this context, "run time" refers to the time after which the protocol or system is initially deployed, and thus differs from real-time constraints in order to meet service requirements.

During design time, the protocol designer should ensure that all major stakeholders are identified and their interests are taken into account. This task requires an open-minded view in order to include all roles that are affected by a tussle. It is important to have in mind that stakeholders may constantly change, for example new ones can enter the tussle, and this should be done with minimum spillovers.

After identifying relevant stakeholders and their interests, a protocol designer has to determine the supported actions and who can perform each one of them. These actions form the "interfaces" that allow stakeholders to interact with each other. The goal should be to allow every stakeholder to influence the tussle outcome so that collateral effects are avoided. This means that control should be distributed, even though some stakeholder instances may prefer not to exercise their right. One way to achieve this goal is to build interfaces that are open, which means standardized but at the same time flexible enough to capture unpredicted cases.

We should keep in mind that unless the interests of stakeholders are adverse, the tussle at run-time will lead to a stable outcome. This, for example, can be achieved through economics, or another reciprocative method. As Clark et al. [5] mention, if such a reciprocative method can be found then it should be implemented by following the same procedure recursively. A tussle outcome may be temporary since Internet is not a "closed" engineering system. An event triggered during run-time may tilt the tussle into a new equilibrium. This is perfectly reasonable as long as the tussle is fought out within the 'playground' defined by the tussle space boundary of the protocol.

Clark [5] mentions SMTP as a protocol that is designed for choice. During the configuration phase a user selects which provider will forward the email. However, some ISPs may not like their customers making this choice, and could undertake Deep Packet Inspection during run-time to block the well-known port in order to exert control (i.e. force the usage of their mail servers). It is clear that this is not the way in which a tussle should be played out, since they are applying a brute force method to restrict their customers' choices. We will try to clarify the notion of a protocol that is "*designed for choice*", by explaining why some well-known protocols seem to be compatible to this principle, and some are not.

Perhaps, a more straightforward example is BGP. ISPs are free to devise their own routing policies, but neighbouring providers can express their preferences by using simple BGP mechanisms. In particular, these preferences can be exchanged by using attributes such as Multi-Exit Discriminator (MED) and Communities[1]. These features allow distributed control at run-time. The reason is that ISPs are not restricted to perform shortest path routing based on longest prefix; they have the ability to select routes based on a wide range of criteria.

ENUM is an example of protocols that allow for "variation of outcome". During configuration an end user becomes a subscriber (opts in) and fully controls the level of details to be inserted in the database. For example she could elect to publish all possible ways of contact along with the associated preferences-wishes, or hide her personal addresses. At run-time, the query issuer has the ability to select which contact address will be used for the session setup. In the case of a VoIP call for example, the signaling server is not restricted to follow a destination's preferences; it can apply

[1] Allowed expressions are described following an out-of-bound method (usually manually).

its routing strategy and select the most appropriate contact address(es) to use for any single reason or combination (lower cost, supported signaling protocol, etc). Furthermore, regulator's interests are taken into consideration so that only valid owners of a telephone number can be registered into ENUM.

Staying in the VoIP context, SIP and H.323 are examples of protocols that are designed for choice. The first versions of the H.323 protocol suite were less flexible, since a provider's signaling server (called gatekeeper) had a pivotal role in session setup. For example, a device had to request permission from a gatekeeper for any call attempt, while the latter could deny service if it sensed that network conditions did not meet customer expectations. Since H.323v4 these protocols have converged, for example gatekeepers are optional components, addresses have the same structure, and both support protocol extensions for third party applications. Nowadays, both protocols can be used in a wide range of configurations; from closed systems like IP Multimedia Subsystem (IMS) to end-user installations (i.e. OPENSIPS, OpenH323). A signaling server (of either protocol) may redirect the calling party towards the destination, may act as a proxy only for signaling, or participate in both signaling and media path in order to take advantage of MPLS networks and comply with regulator requirements (i.e. CALEA). It is important to note, however, that there is no way to influence a signaling server on the way it will handle the request. In case of a VoIP call that can either remain VoIP end-to-end or be set up through a Gateway, then the caller cannot state her preferences.

All TCP variants provide end-to-end congestion control and avoidance by relying on an AIMD scheme that is predefined. This means that unless a user has customized her Operating System kernel, she has no control over the flow's rate. Users, however, have a choice about how many connections they run at any time. This fact has been exploited by peer-to-peer (p2p) file-sharing applications and started a never-ending tussle between ISPs and p2p developers and users (since the former were seeing their links being highly utilized by "some" heavy users) [1]. Even though ISPs tried several means to mitigate their problem, p2p developers could find a counter measure and, again, this resulted in collateral damage to other types of traffic.

NAT is a technology driven by the lack of IPv4 addresses and users' desire for less administrative cost when renumbering their network. In this case control is mainly one-sided; a network administrator deploying NAT has control over the set of incoming connections that are allowed to enter. This is done by NAT pinholes that associate a specific service port to the IP address of a single local host. Care should have been taken, however, so that new protocols are not unfavourably biased. For example, most NAT devices make the assumption that TCP and UDP will be the only transport protocols and do not support newer ones (i.e. SCTP). This fact can stifle future innovation on the Internet due to increased difficulties for a new protocol / service to become widely known and, finally, trusted by users.

In general, it seems that a protocol that distributes control to a number of entities (for example to perform selection or aggregate/disaggregate information, network capacity, etc.) should also allow flexibility in policy used to exercise control, and at the same time should have open interfaces for allowing flexible interaction.

5 Protocol Adoption and Design for Tussle

Balancing traditional engineering and socio-economic goals is very difficult, especially when long-term evolution must be secured, as with the case of Internet. We believe, however, that a protocol being "Designed for Tussle" has more chances in the long-term to be deployed than a protocol that is not. In this section, we present how "designing for tussle" can affect the adoption of previously described technologies.

HTTP is a classic example of a widely adopted protocol. The simplicity, extensibility and layered approach in combination with its clean, modular design, contributed hugely to its success.

BGP is another example of protocol that is "Designed for Tussle". It has modular design and allows distribution of control at run-time in a flexible way. On the other hand, Compact Routing schemes (for example see [12]) try to deal with the problem of routing table memory scalability and provide inelastic routing algorithms. If such a routing scheme was adopted, ISPs would have no control over their routing tables; otherwise, parts of the Internet could be disconnected. This feature is crucial for ISPs and thus compact routing schemes are not expected to be deployed.

In most instances of HTTP and BGP, only two agents are involved and they have enough control to determine the session outcome. But this is not always the case. In VoIP, for example, callers, callees, and providers are only a subset of interested parties; however not all protocols distribute control adequately. Megaco embraces the master-slave paradigm, where all functionality is provided by a signaling server and thus it is not "Designed for Tussle". On the other hand, tussle-awareness and richer functionality of SIP and H.323 gave them an advantage over Megaco. But, the protocol that currently enjoys greater acceptance is SIP, which was standardized inside the IETF. Our feeling is that the main reason is their approach regarding the control distribution between the various stakeholders at their early phases. ITU-based H.323 protocol had many things in common with signaling protocols in circuit-switched networks (SS7), thus control distribution was biased in favour of providers. The better score of SIP in this design principle made it attractive to application developers' eyes who adopted it instead of H.323. Later versions of both protocols converged significantly but it doesn't seem to justify transition to H.323.

It seems that in absence of a protocol that fulfills the criteria of "Designing for Tussle", stakeholders will resort to protocols that provide the highest short-term benefit. Neither NAT nor IPv6, for example, meet the criteria mentioned before; however, the former protocol is widely adopted. The main reason is the fact that NAT is considered beneficial both for the end users and their providers, so they have the incentive to embrace it without considering the long-term consequences. On the other hand IPv6 scores low in functional separation which has a negative impact on backwards compatibility and consequently on providers' incentives to deploy it. However, if IPv6 was redesigned so that it became "tussle-aware" then the outcome could be different in the long term. Providers could gradually move to IPv6 and lessen the need for end-users to turn on NAT devices. Similarly, a "tussle-aware" NAT (for example one that does not restrict what transport protocol is in use) would not harm end-users and, as long as IPv6 is not changed, they would be willing to make a software upgrade to this version.

Another important aspect to consider is the externalities between protocols. Even though a protocol (set) exists that is "Designed for Tussle", its adoption may be delayed until protocols of complementary functionality become tussle-aware. For example TCP is not very modular and provides limited control to users with respect to their sending rate. On the other hand, some of the tussles could be played out independently of each other if users a) were free to select their sending rate, and b) were accountable for the congestion they have caused to other users (i.e. increased delay due to packet loses and consequent retransmissions) given sufficient and timely information about network conditions. This would be possible by using, for example, Re-feedback [2] and a modification of TCP that is able to adjust rate according to user preferences (for example [7]). However, the existence of tussle unaware protocols in the Internet (for example NAT) creates hurdles for the adoption of the more flexible ones, even if they perform different functionalities. As more and more protocols become tussle-aware the pressure to replace bottleneck protocols will be greater and these hurdles will ultimately disappear.

Similarly, although Public ENUM scores high in "Designed for Tussle" criteria, it has seen very limited adoption. Of course, retail VoIP services only recently started to gain significant market share, but it seems that costs and benefits are not aligned across stakeholders. User registration is optional but it assumes that the utility of being reached through the most preferred interface is higher than the registration fee. However, not all VoIP providers accept toll-free calls from other providers because they would like to be compensated for their effort. Thus callers (or their providers) see little benefit from querying ENUM. The fact that an increasing number of providers enter into closed ENUM systems, benefiting from toll-free calls between customers of peered VoIP, gives evidence that adoption of Public ENUM is a matter of supporting economic mechanisms that will align costs and benefits of stakeholders.

Of course, designing tussle-aware protocols and complementary mechanisms increase complexity. Care must be taken to balance technical objectives, such as performance, with socio-economic goals in order for the complexity to be manageable. This could be achieved by capturing the most important factors of stakeholder relationships, without following necessarily the "millions of options" approach [3]. But, we believe that long-term evolution of Internet is more important and this extra cost will be out-weighted by higher functionality and flexibility.

6 Conclusions

This paper has outlined a way forward in designing for tussle by describing a number of important design goals applicable to the architectural evolution of today's commercial Internet. The design principles proposed by Clark et al. have been analyzed using selected examples from the various case studies performed. The isolation of tussle, through both functional and stakeholder separation, and the design for choice remain fundamental design goals.

We can conclude that "designing for tussle" does exhibit benefits when designing new protocols, but it is not sufficient condition to ensure the short-term success of a certain protocol, system or technology. Some technologies – whilst designed for tussle – have not been successfully deployed and adopted immediately, while others

have been very successful – despite not being designed for tussle. However, we believe that tussle-aware protocols are very important for the long-term evolution of Internet. Last but not least, care must be taken to balance technical objectives, such as performance, with socio-economic goals so that the complexity is manageable.

Acknowledgments. The authors wish to thank all participants of the Trilogy project who contributed to this work through discussion, review, and case study. In particular, the authors would like to thank Simon Schütz, K. Richardson and R. Widera for their initial contribution. C. Courcoubetis and J. Crowcroft also provided useful comments and feedback.

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

Costas Kalogiros was also financed by E.U.-European Social Fund (80%) and the Greek Ministry of Development-GSRT (20%).

References

1. Briscoe, B.: Flow Rate Fairness: Dismantling a Religion. ACM SIGCOMM Computer Communication Review 37(2), 63–74 (2007)
2. Briscoe, B., Jacquet, A., Di Cairano-Gilfedder, C., Salvatori, A., Soppera, A., Koyabe, M.: Policing Congestion Response in an Internetwork Using Re-feedback. In: Proc. ACM SIGCOMM 2005, CCR, vol. 35(4), pp. 277–288 (2005)
3. Clark, D.D., Sollins, K., Wroclawski, J., Faber, T.: Addressing Reality: An Architectural Response to the Real-World Demands on the Evolving Internet. In: Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture, pp. 247–257 (2003)
4. Clark, D.D.: The Design Philosophy of the Darpa Internet Protocols. In: Proc. ACM SIGCOMM, Vancouver, BC, Canada (1988)
5. Clark, D.D., Wroclawski, J., Sollins, K.R., Braden, R.: Tussle in Cyberspace: Defining Tomorrow's Internet. IEEE ACM Trans. Networking 13(3), 462–475 (2005)
6. Ford, A., Eardley, P., van Schewick, B.: New Design Principles for the Internet. In: International Workshop on the Network of the Future (to appear, 2009)
7. Ford, A., Raiciu, C., Handley, M., Barre, S.: TCP Extensions for Multipath Operation with Multiple Addresses, draft-ford-mptcp-multiaddressed-00, Internet Draft
8. Handley, M.: Why the Internet Only Just Works. BT Technology Journal 24 (2006)
9. Kunniyur, S., Srikant, R.: End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks. IEEE/ACM Transactions on Networking 11, 689–702 (2003)
10. Moors, T.: A Critical Review of End-to-End Arguments in System Design. IEEE International Conference on Communications 2, 1214–1219 (2002)
11. Saltzer, J., Reed, D., Clark, D.D.: End-to-End Arguments in System Design. In: Second International Conference on Distributed Computing Systems, pp. 509–512 (1981); ACM Transactions on Computer Systems 2(4), 277–288 (1984)
12. Thorup, M., Zwick, U.: Compact Routing Schemes. In: Proc. 30th annual ACM Symposium on Parallel Algorithms and Architectures (2001)
13. Wischik, D., Handley, M., Bagnulo Braun, M.: The Resource Pooling Principle. ACM SIGCOMM Computer Communications Review 38(5), 47–52 (2008)