Further Particulars – A147

CHANGE

Enabling Innovation in the Internet Architecture through Flexible Flow-Processing Extensions

School of Computing and Communications
Lancaster University

1 Consortium

CHANGE is an EU FP7 project whose consortium comprises 11 partners: Eurescom (DE), NEC Europe Ltd. (UK), Deutsche Telekom (DE), University College London (UK), Lancaster University (UK), Université catholique de Louvain (BE), TU Berlin (DE), TU Bucharest (RO), Dreamlab Technologies (CH), Nextworks (IT) and Universita` di Pisa (IT).

1 Concept

The Internet has grown over the last twenty years to the point where it plays a crucial role in today's society and business. By almost every measure, the Internet is a great success. It interconnects over a billion people, running a wide range of applications, with new ones appearing regularly that take the world by storm. Indeed the Internet is now displacing more mature technologies such as the circuit switched telephone network and conventional television distribution. Quite simply, it does not make economic sense to provide parallel infrastructure for uses such as telephony that have effectively become "niche" applications; instead, their traffic is now being carried on the same general purpose packet switched network as "data" traffic such as email and the web.

It is this great generality that is the Internet's main advantage. It can be argued that the Internet doesn't do any single task terribly well, but it does everything well enough. And in economic terms, "well enough" is what actually matters. The problem though is that the Internet doesn't really do *everything* well enough. It is more true to say that the Internet provides 80% of the functionality for 20% of the cost. The limitations are well-known: the Internet does not provide predictable quality of service, and does not provide a sufficiently robust and secure infrastructure for critical applications. But to provide 100% of the functionality would require 100% of the costs too. In most countries, ISPs are under immense competitive pressures. This makes it almost impossible to introduce radical changes to the network to support very demanding applications. In fact, even if ISPs wanted to introduce a new network architecture, the network effect makes it infeasible:

early adopters pay all of the costs up front, but gain no benefits until other networks also upgrade. Such network effects are responsible for the very slow uptake of IPv6 for example, even though there is general agreement that an IPv6 world would be a better place.

The inevitable consequence is that the way the core network (layers 3 and 4 in the stack) changes is through accretion of point solutions that provide immediate benefits to the organisation paying the bill. One example of a point solution is a firewall. Deployed at a site border, a firewall provides immediate benefit to the hosts within the site, protecting them from a wide range of unwanted traffic. The site network itself also becomes more manageable, as the network manager now has slightly more control over the traffic on his network. In short, a firewall increases the robustness and predictability of the site's network, even though every network operator knows that a firewall is not a complete security solution. Another example of a point solution is the use of deep-packet inspection (DPI) boxes to classify and rate-limit traffic from certain applications. In the UK, for example, DPI has become nearly ubiquitous over the last couple of years, as the fiercely competitive home broadband market has struggled to cope in the face of falling prices and the rise of applications such as BitTorrent that can crowd out conventional applications and make quality of service intolerable for latency-sensitive traffic such as telephony and games.

The deployment of various forms of flow-aware equipment in the network to improve the service seen by important or common applications seems both inevitable and unstoppable. Such technologies do indeed improve the performance of the network. But the downside is that application knowledge is increasingly being embedded within the network. An arms race has emerged, where operators trying to manage their networks deploy equipment to improve service, either by penalizing "bad" traffic or by enhancing "good" traffic. Application writers then strive to make their traffic look "good". This is clearly a race to the bottom, where it becomes increasingly hard to deploy new applications that do not look exactly like existing ones.

A key part of the problem is that the Internet is fundamentally stupid; this is both its great virtue and its key failing. Although an end-to-end transparent IP network allows huge innovation in applications and protocols, it also means that the Internet does not actually know when it is working. Obviously if packets are not moving then something is wrong, but the network can be moving huge numbers of packets while providing no useful service to its customers: an extreme example is a denial-of-service attack that floods a link with so much traffic that no useful work is done. However, even in less extreme cases it is hard to reason about whether the packet–level service being provided is sufficient for the applications using the network, especially given that the applications of today need not be the applications of tomorrow.

The trend is clear: the Internet can be improved by embedding application knowledge in the network to improve robustness, quality of service, and manageability. The problem is that in doing so, we lose the benefits of generality that led to the success of the Internet in the first place. A packet-switched network that can only support applications that look like today's applications is a poor solution – it has all the disadvantages of packet switching (it is

still not as robust or predictable as a dedicated network would have been), together with all the disadvantages of a dedicated network (e.g., higher management and operating costs, inability to support new applications, etc).

It is against this background of increasing demands being placed on the network (VoIP, TV, games), and increasing embedded knowledge in the network, optimizing today's applications at the expense of tomorrow's, that we propose CHANGE.

The CHANGE Concept: Architecture for Innovation

Our goal is simple: to reinvigorate innovation on the Internet. Achieving this goal requires a careful balance of the principled and the pragmatic.

Whether we would like it or not, there is no returning to the original end-to-end transparent Internet architecture. Quite simply there are just too many reasons why processing of data flows needs to be performed within the network, not just in the end-systems. To enable innovation, we need to play to the strengths of both packet-switching and flow processing, rather than being religiously in one camp or the other. *We fundamentally believe in the advantages of a packet-switched Internet*. What are missing are the primitives to introduce flow-processing at selected key points in the network. This flow processing needs to be done not as implicit hacks, but in a way that makes it a first class object in the network. If done right, we believe this will allow operators and application writers to reason about the emergent behaviour of the end-to-end path through such a network.

> We believe the deployment of a general purpose flow-processing architecture is what is required to break the innovation log-jam that has been developing over the last fifteen years. This is the overall goal of the project.

What, precisely, is flow processing? Flow processing is any manipulation of packets where the service given to those packets is different because they are part of a flow of packets. Flows can be of different granularities: a single TCP connection might comprise a flow, but equally all the traffic between two sites might comprise an aggregate flow, or all the Internet telephony traffic traversing a router might comprise a flow. The concept of a flow is thus very general, but what flow processing has in common is that it requires the maintenance of some measure of flow state in the network.

The processing that might be applied to flows is very varied. At one extreme, flow processing might involve providing lower-latency forwarding to traffic in a flow. At the other extreme, it might involve the reassembly of the contents of a packet stream and parsing of application-level content to perform network intrusion detection, or explicit authentication to a site firewall to allow subsequent packets of a flow to proceed.

Enabling Flow Processing

> To enable innovation, it must be possible to support a wide range of flow processing. At the same time we want to allow rapid innovation and deployment of new flow processing primitives so that flow processors are not locked into the applications of today. Essentially this means that anything more than simple packet forwarding should be a software function, allowing quick deployment. Contrast this with the current Internet, where flow processing is almost always performed in special purpose boxes sold by vendors to solve a specific problem.

Several trends have come together recently to transform this general vision from fantasy into what can be a practical reality. First, general-purpose x86 server hardware has become cheap enough and powerful enough for packet processing at rates of up to 20Gbit/s. The combination of PCI-Express, Gigabit or 10-Gigabit Ethernet supporting virtual queuing, CPUs with many cores, and high-bandwidth NUMA systems architectures, has resulted in low-costs systems that have been optimized for high-performance network processing as servers. These machines are equally at home performing network processing of flows. They have also been optimized for virtualization, which allows a single machine to perform flow processing on behalf of more than one organization. Secondly, high-performance Ethernet switch chipsets have become a low-cost commodity item. Such chipsets have flow-processing capabilities: they can typically perform matching and forwarding based on arbitrary combinations of packet header fields, and also support multipath forwarding which can be used for hash-based load-balancing across multiple output ports. Current commodity chipsets can support flow tables containing tens of thousands of flows. OpenFlow [OPENFLOW] takes advantage of these chipsets by providing a common API to control flow processing in these switches.

Combining OpenFlow-style switches with clusters of commodity servers allows powerful and scalable platforms to be built [FLOWSTREAM]. The switch provides the first level of classification, and balances traffic directly across virtual queues on the servers, allowing traffic to be directed to specific CPU cores for more sophisticated processing. Such a hardware platform provides a very powerful, scalable and flexible base on which to build a flow processing system. An enabling goal of CHANGE is to build upon preliminary work on these platforms, with the goal of building flow processing systems that can take full advantage of the flexibility inherent in such a hardware platform. Such a platform has several advantages:

A) The ability to scale up processing by merely adding more inexpensive servers
B) The ability to scale down processing by concentrating load on a few boxes at quiet times to save power consumption
C) The ability to roll out new flow processing functionality at short notice to handle unexpected problems, or take advantage of unexpected opportunities, with only software reconfiguration required.
D) The ability to support a wide range of functionality thanks to relying on general-purpose hardware and operating systems
E) The ability to dynamically shift processing between flow processing servers

F) The ability to concurrently run different kinds of processing on different sets of flows while providing high performance and fairness guarantees

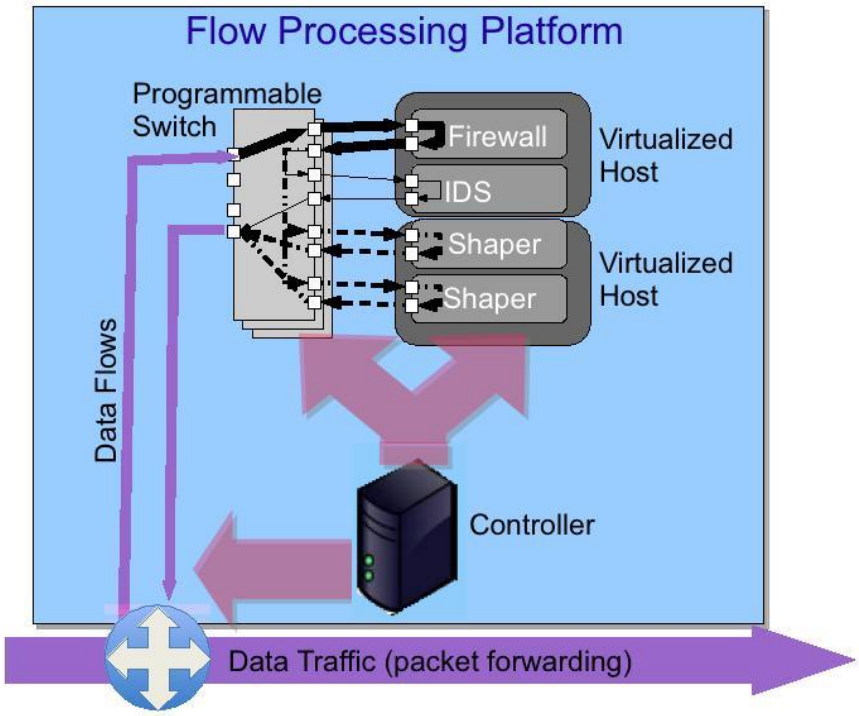

Figure 1 Flow processing platform

Figure 1 illustrates a realization of a flow processing platform, built out of several programmable switches (e.g. OpenFlow switches) interconnecting commodity servers supporting virtualization of processing functions, under the control of the platform controller. Flow processing platforms built in this way have inherent benefits for early adopters, even if no other site deploys them; this avoids the chicken-and-egg problem that affects most proposals for architectural change. However, the goal of CHANGE is not merely to propose an effective way to build end-sites such as data-centers, though that is a first step. The real vision of CHANGE is to build upon this underlying flexible flow processing capability to enable *network-wide* innovation.

A Network Architecture for Innovation

> The broader objective of CHANGE is to use flow processing platforms to enable the Internet to reason about flows and to enhance the processing flows receive in a manner that enables innovation rather than stifles it.

The first step to doing this is to enable rapid deployment of flow processing software into flow processing platforms. This requires that deploying such software can be done safely and in a way that does not adversely impact other traffic using that flow processing platform.

However, we envisage the main advantages will come when flow processing platforms can communicate with each other and with the end systems themselves. For simplicity, it is convenient to consider two different ways in which flow processing platforms can work together to enable innovation:

Virtual networks: Flows can be classified and processed in one flow processing platform, then sent to another flow processing platform for further processing, and so on. Typically this will be done by tunnelling traffic across the Internet. Such flow-level virtual networks give operators and application writers great flexibility in controlling how their traffic is forwarded and where it is processed. This is in stark contrast to the current Internet architecture, where networks process traffic based only on destination address prefixes.

On-path flow processing: Traffic traversing the network using conventional IP forwarding needs to be processed as a flow at certain points in the network; examples are firewalls and traffic shapers. The kind of flow processing platforms we envisage can provide this conventional functionality, but the real benefits come when such flow processing platforms can communicate. This allows applications to express their requirements, networks to express their constraints, and a much more flexible approach to enabling access control restrictions to be taken. Thus an additional goal of CHANGE is to investigate signalling mechanisms by which this communication can occur.

In reality, we envisage these two different concepts, virtual networks and on-path flow processing, to be commonly used in combination, but conceptually they are separate, and the project will address both.

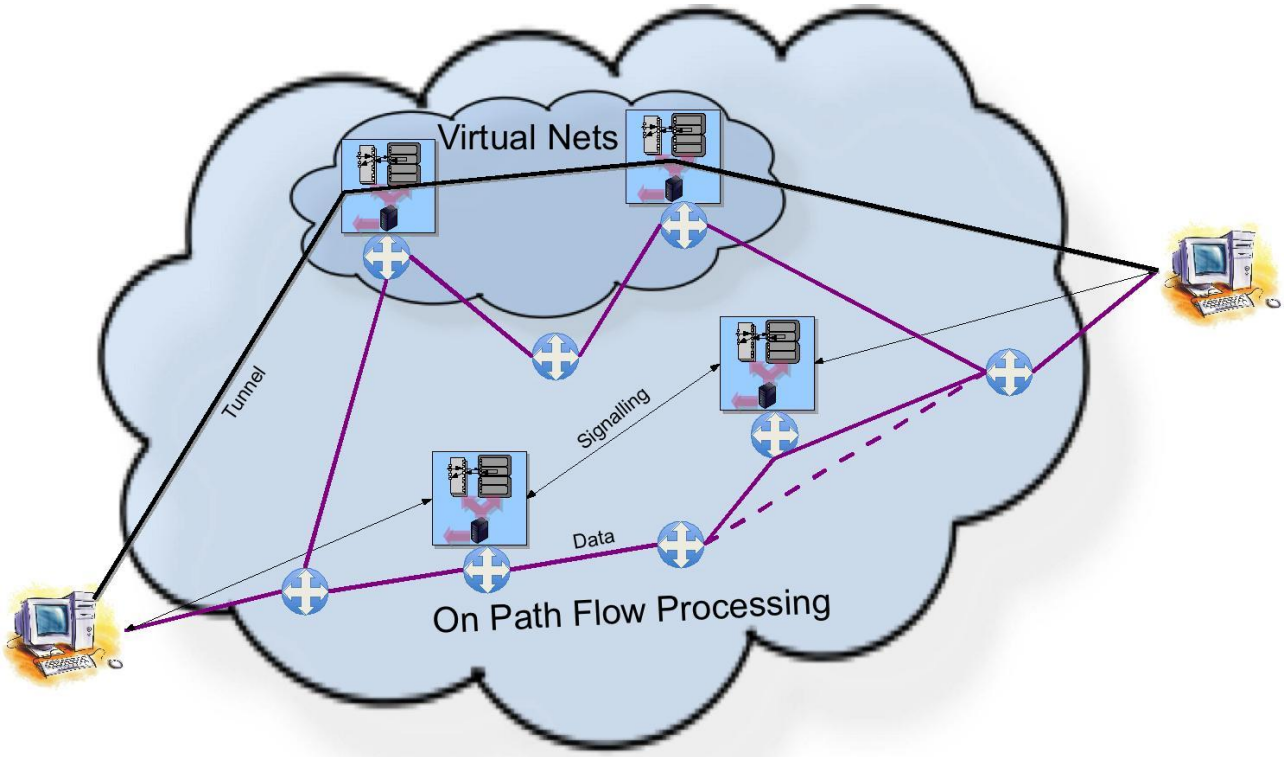

Figure 2 Internet level flow processing

Figure 2 shows some of the possible use of network-wide flow processing. On-path processing uses flow processing platforms very close to the natural IP forwarding path of the packets, essentially coupling with standard IP routers to realize "super nodes". In virtual networks, the flow processing capabilities are overlaid atop the standard Internet; in the case where part of the virtual network is realized on a discrete, separate clean-slate infrastructure, the flow processing platforms can provide an extension of this new infrastructure within the existing Internet, thus greatly extending its reach.

When designing a new technology, especially a technology with as many diverse uses as networking, it would be an act of hubris to believe that we understand all the ways in which that technology will be used. The goal of a network architecture is to allow different requirements to play out differently in different places, while still having a sufficiently rigorous framework of common assumptions. We believe that adding flow processing platforms to the Internet satisfies this balance between enabling new functionality without removing the common assumptions that have made the current Internet so successful.

In the original Internet architecture, in-network processing is limited to IP forwarding. Out of commercial necessity, the current Internet has implicitly extended this to allow various forms of implicit in-network processing, but without adequately defining the place in the

architecture that such processing occupies. We believe that economic pressures will only exacerbate this trend, but the vendors of middle-boxes are ill-placed to take a large enough picture view to extend the architecture itself. In fact no single vendor, or even consortium really is well placed to do this.

> The goal of CHANGE is not to move from one well-defined old architecture to one well-defined new architecture. Rather, the goal is to provide an enabling platform on which new functionality can be rapidly defined, and then to provide sufficient architectural building blocks that give a common frame of reference for future designers.

The intent is that different operators and vendors will each enhance the Internet architecture in their own way, but only for their subset of the traffic. We do not claim insight into all the future ways that the network can be extended using these building blocks; instead, to validate our design choices we will implement a number of our own sample applications that stress the current Internet architecture in different ways.

2 Objective

The CHANGE consortium will collaboratively work towards the following objectives:

1. **Design and specification of an architecture for innovation**
2. **Design and development of flow processing platforms**
3. **Architecture implementation**
4. **Architecture validation through application development**
5. **Dissemination of results**