Demo: Disaggregated Dataplanes

Heena Nagda
Georgia Institute of Technology

Rakesh Nagda
University of Pennsylvania

Nik Sultana
University of Pennsylvania

Boon Thau Loo
University of Pennsylvania

***Abstract*—Modern programmable network hardware enables *in-network computing*—pushing increasingly-complex logic into the network to improve the performance, flexibility and reliability of network services. But the current network programming paradigm is constrained to programming a single network device at a time. The lack of support for in-network programs that use *several* and *heterogeneous* network hardware simultaneously constrains the scale and behaviour of in-network programs.**

***Dataplane Disaggregation* is a new paradigm that addresses this problem. It distributes computations across programmable network hardware including switches and smart NICs. This paradigm transforms a monolithic in-network program into a distributed system executing on possibly heterogeneous resources.**

The goal of this demo is to make an accessible presentation of Dataplane Disaggregation to the wider distributed systems community. This is intended to stimulate discussion on effective ways to program distributed and heterogeneous systems. Our demo is based on the Flightplan system prototype. Flightplan is open-source and comes with detailed documentation and support scripts, yet it requires some effort to set up and run. This impedes its study by others. Our demo runs completely in the browser and does not burden viewers with any installation effort at all.

The technical contribution of this demo consists of a customised visualisation of Flightplan experiments. Moreover, the demo is well-suited to virtual events—as is being planned for ICDCS'21—since it can be run independently and asynchronously by viewers of the demo. This is especially helpful for viewers with slow or intermittent Internet connections. We make the demo's source code freely available online for use by others, including researchers who want to build similar demos.

***Index Terms*—software-defined networking, dataplane disaggregation, distributed systems, network virtualisation, datacenter networking**

I. Introduction

In-network computing [1] leverages the performance and programmability of recent network hardware to run computations in the network. Such hardware is said to have a programmable *dataplane* [2]. Examples of such hardware include programmable switches [3] and smart NICs [4].

A network can have several programmable dataplanes but the current dataplane-programming paradigm is constrained to programming a single dataplane at a time. An in-network program might require more than one dataplane—it might require their combined resources, or the coordination of parts of the network. Therefore the lack of support for in-network programs that use *several* and *heterogeneous* dataplanes constrains the scale and behaviour of in-network programs: either limiting in-network programs to what can be executed on a single dataplane, or burdening the programmer with writing and debugging complex cross-dataplane logic to manually have a program use multiple dataplanes.

To support in-network computing, *Dataplane Disaggregation* was developed to enable programming several, possibly-heterogeneous dataplanes simultaneously. This involves: transforming a monolithic in-network program into a distributed system, exploring different placements for the program's parts in the network, and providing runtime support for fault-handling.

Flightplan [5] provides a prototype implementation of dataplane disaggregation but despite the availability of its source code and extensive documentation, it is laborious to set up. A pre-installed VM would be typically large—tens of gigabytes—and would be opaque to non-specialists.

Demo Goal. The demo seeks to provide an accessible presentation of Flightplan, and aims for the following:

- Rely heavily on concrete examples to make the ideas in the Flightplan paper more widely accessible. The examples are based on experiments featuring disaggregated dataplane programs.
- Engage with a wider community beyond Software-Defined Networking (SDN). The demo aims to stimulate discussion and comparisons with related ideas in other domains and subcommunities.
- Encourage viewer interaction to a degree similar to live tutorials but without burdening attendees with compiling and configuring a research prototype.
- Given that the event is virtual, the demo needs to tolerate asynchrony and delay. This is done by running the demo in the browser and providing documentation and visual cues so viewers can play the demo independently. As a result, the demo will also persist beyond the demo session at ICDCS.

The demo is designed to be simple to use and understand, and can be controlled by the user through the mouse. Fig. 1 shows a glimpse of what is shown to the user, and Fig. 2 provides more details. Users are given an intuitive visualisation of the behaviour of dataplane programs. We built the demo by customizing a platform for teaching and demonstration [6].

The demo is accessible online at https://flightplan.cis.upenn.edu/demo together with supporting documentation.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contracts No. HR0011-19-C-0106, HR0011-17-C-0047, and HR0011-16-C-0056. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

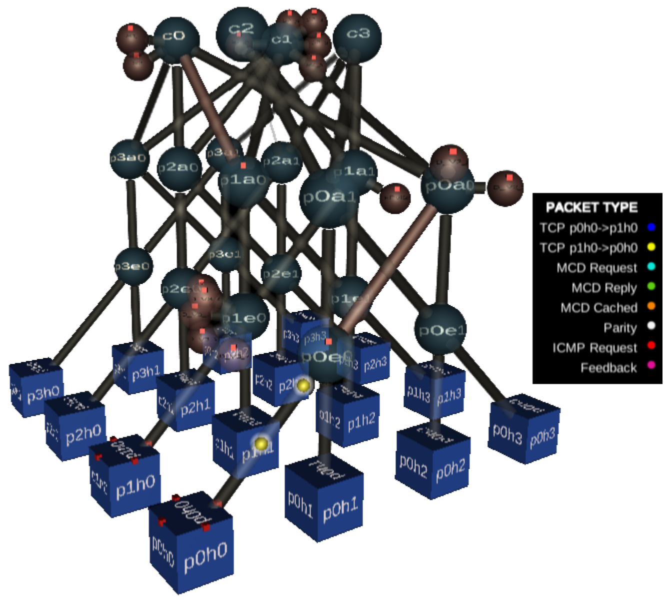


Fig. 1: The demo will consist of a set of experiments in which various dataplane programs process different types of packets flowing through the network. In addition to the topology-based view shown above, the demo will also graph numeric variables of interest to each experiment. Several visual cues are used to help viewers understand the dataplane programs' behaviour. More examples are shown in Fig. 2.

II. Demo Plan

The demo has three stages: **(i)** Introduction: The presenter introduces the demo system, explains how to use it, and shows some example experiments. **(ii)** Viewing: Attendees can run the demo asynchronously in their browser. **(iii)** Interaction: Feedback and suggestions are gathered from attendees, who can also raise discussion points or ask questions about Flightplan or the demo.

III. Demo Content

The demo will feature experiments drawn from the Flightplan paper and code release.[1] These experiments are sequenced to introduce the viewer gradually to different aspects of the demo platform, and presented to the user in the main menu (Fig. 2a).

User Interface. The demo relies heavily on two kinds of visualisation. **(i)** a 3D rendered network topology that shows animated packets traveling through it. **(ii)** 2D graphs that describe quantitative aspects of experiments. These quantitative aspects, such as packet loss, are hard to appreciate by observing the packets travelling in the 3D topology. The demo supports positional and temporal tags that help the viewer understand points of interest throughout the experiments. Positional tags are clickable and provide information about network devices and programs executing in the network.

[1]Flightplan's code is at: https://github.com/eniac/Flightplan

Experiments. The experiments are sequenced to build up to experiment 5 which combines the features encountered in earlier experiments. Experiment 5 is then refined through splitting and offloading to *supporting devices* consisting of other programmable dataplanes. The most complex example is experiment 8 which shows different splits of different programs running simultaneously in the network.

The remainder of this section details each experiment:

1) **Introduction**: This is a tutorial introduction of the demo's interface to the user. During the course of this experiment, tags pop up to explain the functionality of the demo's various features.
2) **FEC booster**: This experiment features the Forward Error Correction (FEC) network booster [7]. Packets are sent over a lossy link in the topology. Some packets are lost while crossing this link. Lost packets are recovered using FEC decoding. Fig. 2d shows a screenshot of this experiment. The graph shows the effect of FEC.
3) **HC booster**: This experiment demonstrates the Header Compression (HC) booster. This booster compresses and decompresses the packet headers. The experiment shows the reduced traffic on the link due to the compression.
4) **MCD booster**: Our Memcached (MCD) booster provides in-network caching of Memcached entries. The experiment demonstrates the cache's role in reducing the traffic load on the Memcached server.
5) **Crosspod**: This experiment demonstrates the `Crosspod.p4` program that combines the FEC, HC, and MCD boosters in one program.
6) **Split Crosspod into 3 parts**: This experiment refines experiment 5 and features the `Crosspod.p4` program split into three parts. The forwarding logic is kept on the switch, while two subprograms containing network boosters are each mapped to supporting devices. The experiment shows packets flowing through the dataplanes that form the distributed program.
7) **Split Crosspod into 6 parts**: This refines experiment 6 into finer-grained spits. In this experiment, the `Crosspod.p4` program is split into six parts. The forwarding logic is kept on the switch, but the booster functions are offloaded to five supporting devices. Fig. 2b shows this experiment's introduction screen, and Fig. 2c shows part of the experiment's execution.
8) **"Figure 7"**: This experiment features the example shown in Figure 7 of the Flightplan paper [5]. It shows various programs split in various ways, and being simultaneously executed in the network. The goal of this experiment was to test complexity and scale.
9) **Fail-over mechanism**: This experiment demonstrates the fail-over mechanism used in Flightplan's 'Full' runtime [5, §5.3]. We simulate the failure of a network device and observe Flightplan failing-over to a backup device. Packets then start flowing through this backup device. The experiment also graphs the packets flowing through the supporting devices over time.

10) **Untunneled traffic**: This experiment provides a baseline for the next experiment (on tunneled traffic). It shows the path followed by the packets in the absence of tunneling, under the influence of routing alone. The experiment is adapted from an example written by third-parties and included in the P4 tutorial [8].
11) **Tunneled traffic**: This experiment shows the effect of the in-network tunneling logic, adapted from a third-party example in the P4 tutorial [8]. Rather than following the routing-defined path, the packets follow the path specified by the configured tunneling.
12) **QoS**: This experiment demonstrates in-network prioritisation based on Quality of Service (QoS), adapted from a third-party example in the P4 tutorial [8]. In the experiment's visualisation, higher-priority packets are shown using a different colour. Using this visual cue we observe the prioritisation of packets changing as they move through the network.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their feedback, and our colleagues for their feedback on different versions of the demo: Haoxian Chen, Max Demoulin, Joel Hypolite, Pardis Pashakhanloo, Lei Shi, Nishanth Shyamkumar, Caleb Stanford, and Ke Zhong.

REFERENCES

[1] T. A. Benson, "In-Network Compute: Considered Armed and Dangerous," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, ser. HotOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 216–224.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming Protocol-Independent Packet Processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, Jul. 2014.

[3] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, p. 99–110, Aug. 2013.

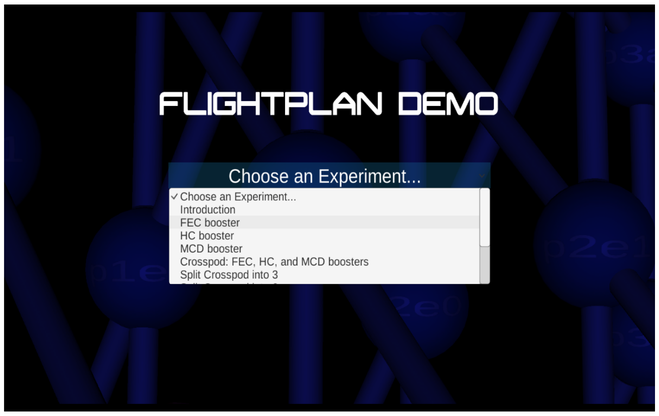
[4] S. Grant, A. Yelam, M. Bland, and A. C. Snoeren, "SmartNIC Performance Isolation with FairNIC: Programmable Networking for the Cloud," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 681–693.

[5] N. Sultana, J. Sonchack, H. Giesen, I. Pedisich, Z. Han, N. Shyamkumar, S. Burad, A. DeHon, and B. T. Loo, "Flightplan: Dataplane Disaggregation and Placement for P4 Programs," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, Apr. 2021. [Online]. Available: https://www.usenix.org/conference/nsdi21/presentation/sultana

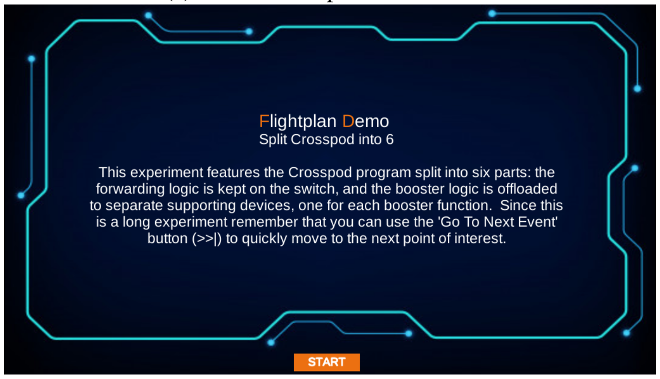
[6] H. Nagda, R. Nagda, I. Pedisich, N. Sultana, and B. T. Loo, "FDP: a teaching and demo platform for SDN," in *CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies, Barcelona, Spain, December, 2020*, D. Han and A. Feldmann, Eds. ACM, 2020, pp. 524–525. [Online]. Available: https://doi.org/10.1145/3386367.3431886

[7] H. Giesen, L. Shi, J. Sonchack, A. Chelluri, N. Prabhu, N. Sultana, L. A. Kant, A. J. McAuley, A. Poylisher, A. DeHon, and B. T. Loo, "In-network computing to the rescue of faulty links," in *Proceedings of the 2018 Morning Workshop on In-Network Computing, NetCompute@SIGCOMM 2018, Budapest, Hungary, August 20, 2018*, X. Jin and C. Kim, Eds. ACM, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3229591.3229595

[8] "P4 Tutorial," https://github.com/p4lang/tutorials, accessed April 2021.



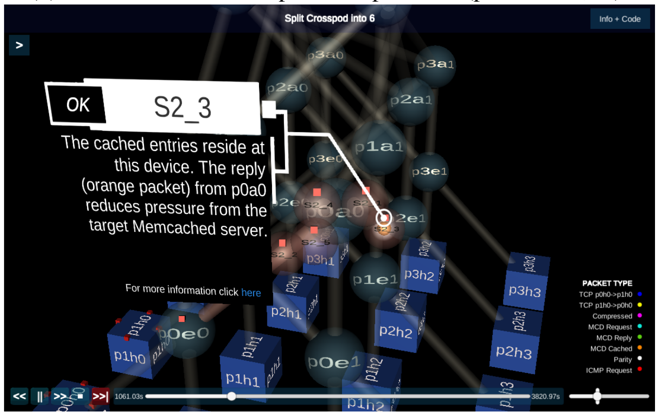
(a) Choice of experiments to run.



Flightplan Demo
Split Crosspod into 6

This experiment features the Crosspod program split into six parts: the forwarding logic is kept on the switch, and the booster logic is offloaded to separate supporting devices, one for each booster function. Since this is a long experiment remember that you can use the 'Go To Next Event' button (>>|) to quickly move to the next point of interest.

(b) Intro screen to the "Split 6" experiment (point 7 in §III).

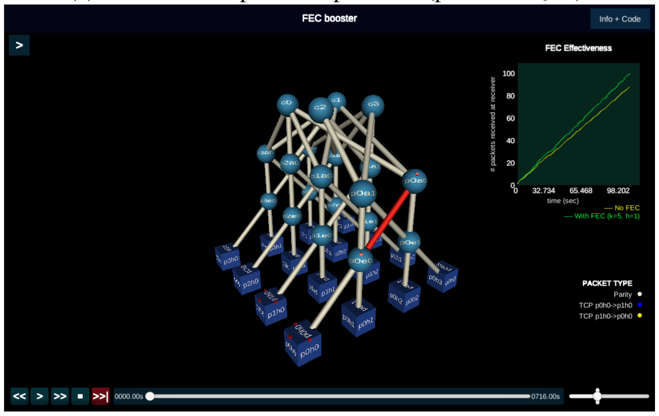


OK S2_3

The cached entries reside at this device. The reply (orange packet) from p0a0 reduces pressure from the target Memcached server.

For more information click here

(c) Part of the "Split 6" experiment (point 7 in §III).



(d) Part of the FEC experiment (point 2 in §III).

Fig. 2: Screenshots from the demo