Interactive Control over a Programmable Computer Network using a Multi-touch Surface

Rudolf Strijkers[1,2], Laurence Muller[1], Mihai Cristea[1], Robert Belleman[1]
Cees de Laat[1], Peter Sloot[1], and Robert Meijer[1,2]

[1] University of Amsterdam, Amsterdam, The Netherlands
[2] TNO Information and Communication Technology, Groningen, The Netherlands
{l.y.l.muller,m.l.cristea,
r.g.belleman,delaat,p.m.a.sloot}@uva.nl
{rudolf.strijkers,robert.meijer}@tno.nl

Abstract This article introduces the Interactive Network concept and describes the design and implementation of the first prototype. In an Interactive Network humans become an integral part of the control system to manage programmable networks and grid networks. The implementation consists of a multi-touch table that allows multiple persons to manage and monitor a programmable network simultaneously. The amount of interactive control of the multi-touch interface is illustrated by the ability to create and manipulate paths, which are either end-to-end, multicast or paths that contain loops. First experiences with the multi-touch table show its potential for collaborative management of large-scale infrastructures.

Key words: programmable network, network management, multi-touch interfaces

1 Introduction

To enable network transparent end-user connectivity, routing protocols or application-specific paths need to be configured by the network operator. However, networks are often complex infrastructures capable of delivering the same service (e.g. an end-to-end connection) in countless ways. Where various degrees of Quality of Service (QoS) are required, the configuration of all the individual devices or network management system becomes a complex task. In addition, routing services handle only basic forwarding and the effects of deteriorated or failing links. In cases for which end-to-end routing services do not apply, or for cases in which the service of the network moves outside the QoS window, a network operator has to take measures to implement the required service or QoS.

In the context of programmable networks, active network implementations [17] and TINA [5] provide applications the tools to configure the network elements (NE), but lack the ability of translating a higher order specification into detailed behavior of NEs. Higher order specifications of the network service can have many forms. A setup-file or a domain specific language is one extreme. In this paper the other extreme is explored:

a dedicated human-network interface in the form of a multi-touch Graphical User Interface (GUI) that supports (1) a direct interface where actions in the form of gestures automatically translate in manipulations on individual network elements, (2) real-time, direct interaction with the network that graphically represents the results of actions and (3) collaborative control-room applications where multiple persons manage or monitor a programmable network simultaneously.

The Interactive Network concept was demonstrated at the Dutch research exhibition booth at Super Computing 2008 (SC08), Austin, Texas. The network contained servers with multiple High Definition (HD) video streams that could be routed to HD screens. The demo setup allowed multiple visitors simultaneously to create, manipulate and remove video streams by the touch of fingers.

The next section provides a summary of related work, both for the networking part and the GUI. Section 3 presents the concepts and overall architecture. In section 4, we describe Interactive Network prototype features and its implementation. In section 5, preliminary results are presented. Section 6 gives insight in the current issues, pointers to future work and concludes the article.

2 Related work

TINA is one of the first efforts that applied IT accomplishments into the domain of networking. One subproject, ACTranS [3] investigated distributed transaction processing support for networks, which resulted in an Object Management Group [15] compliant architecture for transactional path and connection setup services. In addition, the ACTranS Tutorial Demo was a showcase for setting up, manipulating and removing ATM paths with a GUI. However, ACTranS targeted only ATM technology and did not support more than end-to-end connectivity. Furthermore, the complexity of the architecture and implementation never led to adoption by industry.

Nowadays, large-scale efforts in optical and hybrid networking such as UCLPv2 [8] and GEANT [1] use service oriented architectures. By exposing the network element functions as web services these networks support application programmable light paths, and also offer GUI for path management. Among other GUIs for resource management are HP Openview, VMware Infrastructure Manager and workflow managers, such as WS-VLAM [20] to control grid services. However, all the GUIs remain desktop-based single-user applications for network resource provisioning.

With the introduction of the camera based multi-touch technique called FTIR that Jeff Han [9] presented in 2005, multi-touch systems became affordable for research projects. In comparison to traditional input devices (mouse, touchpad and touch screen), multi-touch systems allow multiple points of interaction simultaneously. Complex interactions that require multiple steps in single touch input systems can be simplified with gestures using multiple fingertips.

Until now multi-touch systems were mainly used for scientific visualization and entertainment projects. Most of these projects are related to one or more of the following categories: photo and video organization, paint applications, fluid simulation [13] or geographic information systems [7]. Recent projects show new applications in the field of graph visualization. These projects are often based on existing applications [10] and are trying to visualize, often static, social or computer networks in order to find common structures or properties. By using multi-touch it becomes possible to view and navigate through large data sets. However, most of these applications share one common limitation: the application only allows the user to control the view and the layout of the data set. In other words, it is not possible to modify data structures.

3 Concepts and Architecture

Programmable networks are more flexible than traditional networks and offer advantages when more than best effort end-to-end services are needed. However, the task of managing a programmable network is different from normal IP networks. Other than IP networks, there is not a single best effort end-to-end service, but a collection of programmable components that are loaded into network elements by default or which can be uploaded on demand. Hence, the resources in a programmable network need to be coordinated, either by hand or by programs that automate decision processes, such as default shortest path routing for applications, or application-specific paths at application request.

Figure 1 shows the User Programmable Virtualized Network (UPVN) [14] architectural framework, which defines the primary components that apply to all programmable networks. Network elements (NE) contain loadable software objects called application components (AC) and allow implementation of not foreseen or application-specific behavior. NEs can also provide a set of default ACs. ACs allow computer programs to access their service interfaces through network components (NC), which are components embedded in applications that interface with the network. NCs act as proxies that provide redirection, virtualization or composition of AC service interfaces. The manner in which NCs are exposed to applications is application-specific, but also the implementation of communication middleware depends on the application domain. For example, monitoring interfaces might be asynchronous and event-based, while control interfaces that require direct response will be synchronous.

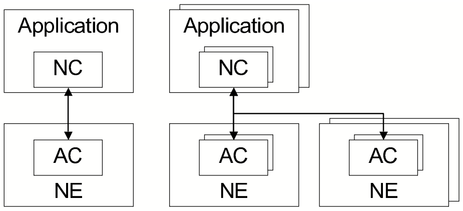


Figure 1. Relating the UPVN components. Network Components (NCs) are the manifestation of the network in applications. Conversely, Application Components (ACs) manifest as application-specific network services. Implemention choices for NC and AC communication depend on the application-domain

Interactive management of a UPVN requires (1) visualization of network state and (2) interaction methods that allow manipulation of the software components loaded in the network. The human becomes part of the decision process by utilizing network visualization for sense making and using interaction methods to persist decisions. Figure 2 shows the relationship between the components that make up an interactive network, which is composed of four elementary components.

Network Resource Control The services provided by individual NEs determine the finest element of detail that can be controlled (Figure 2, 1). Services in a NE implement specific adapters or low-level control-loops over NE functions (Figure 2, 2).

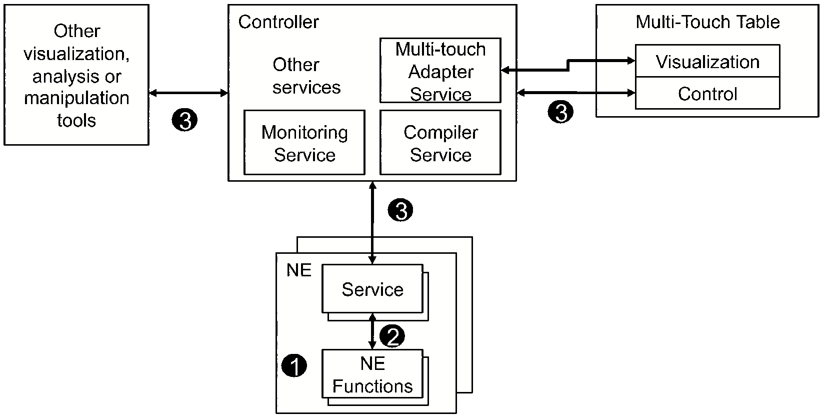


Figure 2. The architectural components of an Interactive Network: humans interact with the network and have become part of the control loop.

A controller application implements services that span over multiple NEs, such as distributed transaction processing or maintaining network topology. The controller application includes the adapter between interaction tools, visualization and human control and for compiling requests into network manipulations.

Middleware The system has to agree on protocols, services that network elements deliver and their communication mechanisms (Figure 2, 3). Individual services can be discovered and consumed by applications or controllers in the network when implemented by the middleware.

Visualization The visualization of network state must be in such a manner that it becomes comprehensible to a human and provides useful information for decision support. Visualization of the network spans multiple levels of detail, such as forwarding expressions and rules, routing and inter-domain connections.

Graphical User Interface The GUI implements interaction with any part of the visualized network. To be suitable for collaborative environments the GUI should support multiple users at the same time that interact with the network. A multi-touch table can provide a suitable environment, because, other than traditional point and click devices, users can gather around the table and use the same GUI.

4 Implementation

We have built an Interactive Network prototype that implements the presented architecture. Figure 3 shows the setup presented in a live demo at the Dutch research exhibition booth at SC08. The test bed consists of three VMware ESX [18] servers each containing four virtual machines (VM) (Figure 3, 1) running Linux and interconnected by a virtual switch, two commodity Linux servers (Figure 3, 2) and four Mac mini's. A physical gigabit switch connected the VMs and other machines. The Mac mini's were connected through a dedicated path from the booth to our test bed in Amsterdam. There was a separate network (dashed lines in Figure 3) to which the multi-touch table was connected. This allows the programs on the multi-touch table and on the nodes to communicate with the control software, without interfering with the manipulated data streams.

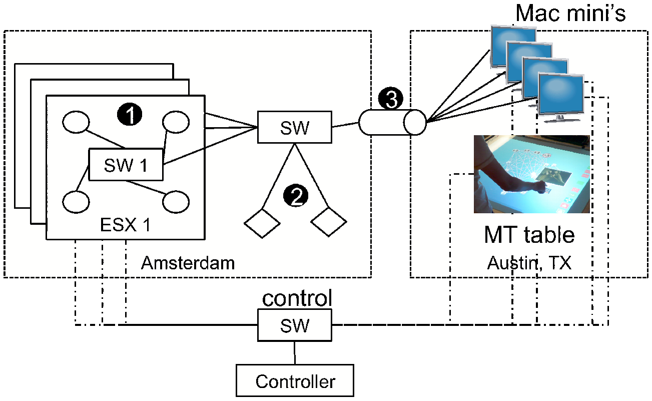


Figure 3. Hardware setup of the programmable test bed.

All the Linux nodes run an AC implemented in Java that provides an interface to local services. Local services implement adapters to among others, Nmap [2] an application to discover neighbouring network elements and Streamline [4], an application that manipulates network traffic in the Linux kernel. A controller application runs on a separate node, which remotely orchestrates behavior of the ACs. The address of this node is known to all nodes part of the network. At boot time, the ACs connect to the controller, which then sends a discovery request to the client to find neighbouring NEs. The controller determines the known topology by combining discovery information of all the ACs, which is then sent to and displayed by the GUI (Figure 4).

Sensing and tracking multi-touch on the table is handled by a separate library, Touchlib [19]. The multi-touch GUI is implemented in Actionscript 3 and runs in the Adobe AIR runtime environment [11]. The implementation supports three different interaction modes: routing mode, local stream manipulation mode and a viewing / monitoring mode.

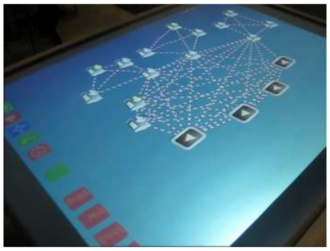


Figure 4. Test bed as visualized on the multi-touch table.

Routing In routing mode users can define streams from producers to consumers. New streams are defined by dragging a line from node to node, starting with a producer and ending with a consumer (Figure 6(b)). On desktop systems, the most common method is to click the nodes in sequence of the path to create. While it is possible to adopt this method for multi-touch, it also introduces a single-user limitation. Touch and drag can also be used in multiuser environments, such as the multi-touch table. At touch, a stream is assigned a unique color, which is also used to identify the traffic in the network. This is achieved by encoding the color id in the IP options field of the IP traffic when provisioning the programmable network. The system supports three kinds of paths:

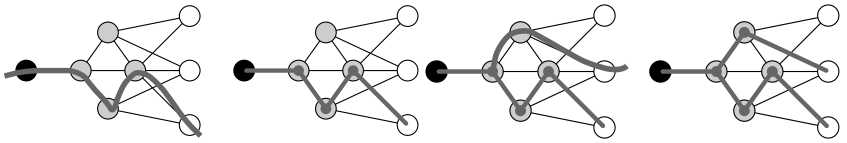
- Point-to-point drawn from producer to consumer (Figure 5(a)).
- Multicast, drawn by extending an existing point-to-point stream (Figure 5(b)).
- Point-to-point containing a loop. This stream contains a path that crosses a specific node multiple times (Figure 5(c)). We added the ability to create loops to illustrate that some operations, intuitive to a user or application, are not intuitive at all from the perspective of a network. UPVN accommodates such unforeseen application demands, where traditional approaches cannot.

The controller implements a distributed transaction processor to support atomic operations over multiple nodes, such as creating paths. In the case any of the nodes fail while provisioning a path, the whole operation is rolled back and the drawn path is removed from the visualization.

Local stream manipulation After creating a path in routing mode, users can modify the expressions generated by the path compiler on any node. By double tapping a node, an overview of currently active streamline expressions is displayed in a zoom window. This way it is still possible to maintain a global overview of the network and still allow other users to interact with other parts of the network.

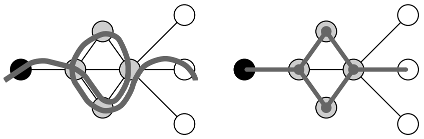
Streams can be distinguished based on the color assignment at creation. The streamline expressions show the pipeline of filters that are applied to incoming packets before they are sent out. Each filter in the streamline expression represents an application component implemented in a kernel module.

A streamline expression illustrated as a graph can also be modified (Figure 6(c)). Currently, the application only supports adding and removal of sampler filters. Sampler filters allows users to adjust the packet drop rate of a stream. To add a rate limiter to a stream, the user touches a node in the graph and taps the plus sign. This



(a) Creating a point-to-point path.

(b) Extending a point to point path to a multicast path.



(c) Creating a point-to-point path containing a loop.

Figure 5. Examples showing how to create a path or modify an existing path in the application. The black node is a producer, the grey node a router and the white node a consumer.

action will send a request to the controller, which will delegate the request to the correct node to update its streamline graph.

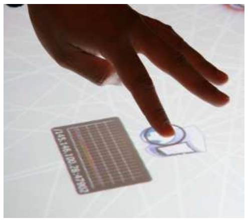
Viewing / Monitoring When multiple users can gather around the multi-touch table, each person will view the display from a different orientation. This becomes an issue in text visualization, which is strongly oriented towards the viewpoint of a user. To avoid this issue, we chose to use as much icons as possible to represent the various node types and GUI elements. An other solution to this issue would be to set default orientations were users can view the multi-touch GUI.

Apart from the network specific interactions, the panel also supports display of monitoring data and manipulations on the network visualization. In viewing mode, for example, the current load on a node can be viewed by touching the icon (Figure 6(a)). A small graph appears that shows information of the current workload and the average workload over the past 5 and 15 minutes.

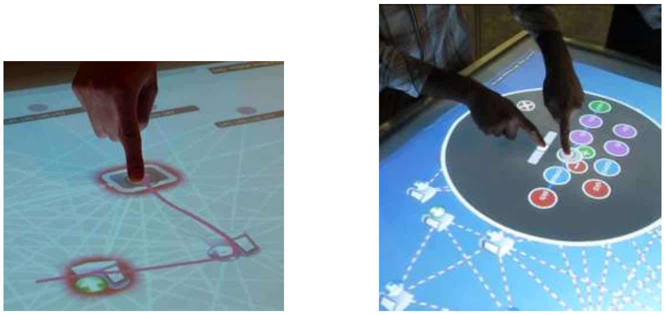
In viewing mode users can also navigate through the network by using the touch of their fingers. Gestures such as dragging and pinching, allow the user to rapidly move the view or zoom into the network. When visualizing a large number of nodes, it becomes difficult to maintain an overview of the network. Therefore, the multi-touch GUI also allows a user to activate automatic layouting. This layout algorithm is provided by the flare library [12].

5 Results

The Interactive Network prototype at SC08 received positive feedback. Visitors were invited to play with the Interactive Network. Multiple users could interact with the system and see the results of their manipulations on the four screens. The direct feedback in terms of video playback proved valuable to help non-network experts to use the system without requiring knowledge of the underlying programmable network.



(a) Monitoring the router node workload.



(b) Creating a new path from a producer to a consumer.

(c) Adjusting a streamline rate modifier.

Figure 6. Examples of different interaction layers of the application.

The robustness of the entire test bed was assured by the controller application, which has no a priori knowledge about the nodes that are part of the network. Hence, failure of nodes did not affect the system as a whole. The prototype remained running for four days of testing by visitors without requiring restarts, and was robust against failure or restarting of individual nodes.

The setup also had some limitations. First, due to the multi-touch tracking mechanism environment light had an impact on its accuracy. Second, currently it is not possible to drag paths and manipulate stream samplers at the same time, because the routing and stream manipulation modes apply to the whole interface. Although in practice this did not pose any problems, for more complex systems it is preferable to support multiple types of interaction at the same time to different parts of the GUI.

6 Conclusion and Future work

The Interactive Network prototype was designed as a command-and-control environment for programmable networks. By using multi-touch technology we have enabled multiple persons to manage a programmable network simultaneously, without requiring expert knowledge on networks.

Compared to desktop applications, multi-touch devices allow direct interaction with visualization. By controlling the network using multi-touch, management tasks over many network elements can be simplified. When users can have an overview of the system at all times, configuring or programming a collection of network elements is less error prone compared to typing in commands in a terminal for each device separately. However, comparative research is needed to determine the usefulness of multi-touch GUIs compared to traditional point and click devices or collaborative web applications. For example, can the use of multi-touch systems increase the effectivity of collaboration in case of extreme situations, such as sudden calamities in a grid infrastructure or a complex management tasks? In addition, multi-touch enabled automation tools and software interfaces, such as workflow managers, domain specific (visual) programming languages or schedulers will be needed to support large-scale grid and application management and comparison to the traditional approaches.

The Interactive Network concept can also be applied to other domains, such as Interactive optimization of data centers, include grid services or to interact with computational models that run on grids. The OptiPuter [16] research project, for example, presents a vision of interactive access and control to high-speeds networks, large data-sets and high-performance computing clusters to support e-Science. If the computational and network resources are abstracted and enriched with meta-data, by using NDL [6] for example, multi-touch GUIs can improve the way scientists interact with computational and infrastructural resources to coordinate their experiments and view its results.

Future efforts will include adding semantic information about the network and its resources and include control over more than network services alone. Furthermore, to control grid resources and orchestrate e-Science experiments with multi-touch interfaces will also need research into visualization aspects, such as zooming user interfaces, and intuitive multi-touch gestures.

Acknowledgments

We gratefully acknowledge the help of Willem de Bruijn, Paul Melis, Edwin Steffens and Edward Berbee. This work was supported in part by the European Commission under the project ACGT: Advancing Clinicogenomic Trials on Cancer (FP6-2005-IST-026996) and in the context of the Virtual Laboratory for e-Science (VL-e) project (http://www.vl-e.nl). The VL-e project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation programme of the Ministry of Economic Affairs (EZ). Part of this project was also funded by GigaPort, Phosphorus and the European Union (EU) under contract number 034115. We also thank the Netherlands Organization for Scientific Research (NWO) for their support and organization of the Dutch research exhibition booth at SC08.

Bibliography

[1] *GEANT network*, `http://www.geant.net`.

[2] *Network mapper*, `http://nmap.org`.

[3] ACTS, *A transaction processing toolkit for acts: Actrans, final report*, (1998).

[4] Herbert Bos, Willem de Bruijn, Mihai Cristea, Trung Nguyen, and Georgios Portokalidis, *FFPF: Fairly fast packet filters*, OSDI (2004).

[5] Martin Chapman and Stefano Montesi, *Overall concepts and principles of tina*, Tech. report, TINA Consortium, February 1995.

[6] Freek Dijkstra, Jeroen J van der Ham, Paola Grosso, and Cees de Laat, *A path finding implementation for multi-layer networks*, FGCS **25** (2009), 142–146.

[7] Clifton Forlines, Alan Esenther, and et al, *Multi-user, multi-display interaction with a single-user, single-display geospatial application*, Proceedings of the 19th Annual ACM Symposium on User interface Software and Technology (New York, NY, USA), ACM, 2006.

[8] E. Grasa, G. Junyent, and et al, *UCLPv2: A network virtualization framework built on web services*, IEEE Communications Magazine **46** (2008).

[9] Jefferson Y. Han, *Low-cost multi-touch sensing through frustrated total internal reflection*, Proceedings of the 18th Annual ACM Symposium on User interface Software and Technology (New York, NY, USA), ACM Press, 2005.

[10] Jeffrey Heer and Danah Boyd, *Vizster: Visualizing online social networks*, Proceedings of the 2005 IEEE Symposium on Information Visualization (Washington, DC, USA), 2005.

[11] Adobe Inc., *Adobe AIR*, (2008), `http://www.adobe.com/products/air/`.

[12] UC Berkeley Visualization Lab, *Flare: Data visualization for the web*, (2008), `http://flare.prefuse.org/`.

[13] Jefferson Y. Han, *Multi-touch interaction wall*, ACM SIGGRAPH 2006 Emerging Technologies (New York, NY, USA), ACM Press, 2006.

[14] Robert J. Meijer, Rudolf J. Strijkers, Leon Gommans, and Cees de Laat, *User programmable virtualized networks*, Proceedings of the Second IEEE international Conference on E-Science and Grid Computing, 2006.

[15] The Object Management Group (OMG), `http://www.omg.org`.

[16] Larry Smarr, Maxine Brown, and Cees de Laat, *Editorial: Special section: Optiplanet - the optiputer global collaboratory*, FGCS **25** (2009), 109–113.

[17] David L. Tennenhouse and David J. Wetherall, *Towards an active network architecture*, SIGCOMM Comput. Commun. Rev. **46** (2007).

[18] VMware Inc., *VMware*, `http://www.vmware.com`.

[19] David Wallin, *Touchlib: an opensource multi-touch framework*, (2006), `http://www.whitenoiseaudio.com/touchlib/`.

[20] A. Wibisono, V. Korkhov, and et al, *WS-VLAM: Towards a scalable workflow system on the grid*, Proceedings of the 2nd Workshop on Workflows in Support of Large-Scale Science (Monterey Bay, California, USA), June 2007.