A Virtual Laboratory for Distributed Systems Research

Examples in Research (MAKI) and Teaching

ICSI, Berkeley
March 31, 2015
BMBF project 2008 – 2012

Closing the loop between research and real-world experiments

Provide an experimental facility for studies on architectures, mechanisms, protocols and applications towards Future Internet

Investigate interdependency of theoretical studies and prototype development
Project G-Lab

- WP0: Coordination and Consolidation
- WP1: Architecture of the Future Internet
- WP2: Routing and Address Schemes
- WP3: Wireless Networks and Mobility
- WP4: Monitoring- und Management Concepts
- WP5: Quality of Service and Security
- WP6: Architecture and Composition of Services
- WP7: Experimental Facility
- Other projects
The Lab

- Distributed resources across 6 sites
- Central administration at Kaiserslautern
- Connectivity German NREN (DFN)

Exclusive resource reservation
- Decentralized Resources can be independently used
- Tests on the lower layers of the network without affecting the operating network

Control frameworks
- PlanetLab (PLC at Kaiserslautern)
- ToMaTo
- Interfaces to other testbeds (GENI, PlanetLab Japan, WinLab, …)
- OpenFlow setup (virtual as well as hardware setup)
**Tomato / G-Lab Testbed**

- **Tomato Topology Management Tool**
  - A part of German-Lab research project WP7
    - Open Source since version 2.0 (hosted at github)
    - Can be used independently from G-Lab
  - Currently >12 partner institutions (worldwide)
  - >122 Nodes: up to 256 GB memory, 4-24 cores
  - A topology-centric network testbed & virtualization
  - Each experiment has its own virtual network topology + virtual hosts; each topology is self contained
  - WebUI
    - Intuitive topology definition and management
    - Direct access to virtual machines: e.g. VNC console
  - Advanced features for topology management:
    - JSON-based description language (import/export of topologies)
    - Python XML-RPC API (also used by WebUI itself)
Topology- vs. Slice-Based Orientation

- **Tomato** is topology-oriented
  - Basic abstraction is the topology
  - Hosts provide virtual components
  - Topologies can be constructed using those virtual components
  - Tunneling and stitching is inserted automatically
  - Not restricted by physical topology

- PlantLab/GENI facilities are slice-based
  - Slice is a subset of the existing topology
  - Networks are isolated by VLans
  - Topologies are restricted by physical topology

- Practical differences
  - Fully virtual network components in **Tomato** (routers, hubs, switches, OpenFlow, etc.)
  - Automatic layer 2 tunneling, no need for physical layer 2 connections
  - Ability to use multi-homing even when no site is multi-homed
Framework Comparison

- Repy
- Seattle
- OpenVZ
- PLANETLAB
- KVM
- TcMaTo

Parallellism vs. Realism
## Framework Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Planet-Lab</th>
<th>Emulab</th>
<th>Seattle</th>
<th>Tomato</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple sites</td>
<td>✔</td>
<td>☐</td>
<td>(✔)</td>
<td>✔</td>
</tr>
<tr>
<td>Physical hardware access</td>
<td>☐</td>
<td>✔</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>End-System virtualization</td>
<td>✔</td>
<td>(✔)</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Network virtualization</td>
<td>☐</td>
<td>✔</td>
<td>☐</td>
<td>✔</td>
</tr>
<tr>
<td>Layer 2 access</td>
<td>☐</td>
<td>✔</td>
<td>☐</td>
<td>✔</td>
</tr>
<tr>
<td>Link emulation</td>
<td>☐</td>
<td>✔</td>
<td>☐</td>
<td>✔</td>
</tr>
<tr>
<td>Packet capturing</td>
<td>☐</td>
<td>(✔)</td>
<td>☐</td>
<td>✔</td>
</tr>
<tr>
<td>High traffic (&gt;100 Mbit/s)</td>
<td>☐</td>
<td>✔</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Resource profiles</td>
<td>✔</td>
<td>☐</td>
<td>☐</td>
<td>✔</td>
</tr>
<tr>
<td>VNC control</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>✔</td>
</tr>
</tbody>
</table>
Hostmanager
- Hosts based on Proxmox VE
- Controls one host,
- Offers virtualization/network capabilities
- Controls local topology elements

Frontend(s)
- Multiple frontends possible
- Currently: Web-based, CLI

Backend
- Controls whole topologies
- Distributes topologies over hosts
- Applies stitching
- Delegates management to hosts
- Monitoring
- User management
- Provides XML-RPC interface

- Command line interface (API)
- scripting
Tomato - the elements

- Topology contains
  - Devices: produce and consume data; can run software
    - Three kinds of devices
      - KVM devices (green)
      - OpenVZ devices (blue)
      - Programmable devices (orange)
  - Connectors forward and manipulate data and connect devices
    - Two kinds of connectors
      - VPN networks (based on Tinc)
      - External networks

- Graphical representation
  - Icons show element type
  - Colored icons show virtualization technology
  - Link color shows network segments
  - Link style shows link attributes

- Per Topology
  - Accounting
  - Permissions
VM Elements

- **KVM**
  - Full virtualization
  - Integrated into Linux Kernel

- **OpenVZ**
  - Container virtualization
  - Added to Linux Kernel via patch

- **Scripts**
  - Programming language virtualization
  - Installed as software

- **Repy scripts**
  - Restricted Python (Sandbox)
  - Technology from Seattle testbed
  - Modified for Tomato
  - Functions for receiving and sending raw ethernet packages

- **Additional elements**
  - Easy to add more
  - Planned: VirtualBox, LXC, …

<table>
<thead>
<tr>
<th>Features</th>
<th>KVM</th>
<th>Open VZ</th>
<th>Repy scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td># per node</td>
<td>~20</td>
<td>~100</td>
<td>~1000</td>
</tr>
<tr>
<td>any x86 OS</td>
<td>✅</td>
<td>▭</td>
<td>▭</td>
</tr>
<tr>
<td>Linux OS</td>
<td>✅</td>
<td>✅</td>
<td>✭</td>
</tr>
<tr>
<td>Kernel space</td>
<td>✅</td>
<td>▭</td>
<td>✭</td>
</tr>
<tr>
<td>Console support</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Mouse/Keyboard input</td>
<td>✅</td>
<td>✅</td>
<td>▭</td>
</tr>
<tr>
<td>Layer 2 connectivity</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Interface configuration</td>
<td>(✔️)</td>
<td>✅</td>
<td>(✔️)</td>
</tr>
</tbody>
</table>
- Rename the elements
- Choose a site otherwise the element will be distributed based on load balancing
- Profile defines #CPU, RAM, Disk

- Choose a template
  - Linux
  - Windows (on KVM systems)
  - Programmable devices (Repy)
  - Open Vswitch (KVM)
Connectors / Network Elements

- **VPN: Tinc**
  - Full mesh VPN without server
  - Fully contained, virtual network
  - Cross-site layer 2 connectivity
  - Open endpoints allow federation

- **Tunnel: VTun**
  - Layer 2 tunnel over UDP
  - Open endpoints allow federation

- **External networks**
  - Bridge into local network segments
  - E.g. Internet or local research network

- **SDN / OpenFlow**
  - openVswitch / NEC
  - Floodlight / Ryu OpenFlow Controller
Tomato – Features and editor

- Administrator/Developer features
  - Intelligent load-balancing
  - Open xml-rpc interface
  - LDAP integration

- User features
  - Automatic network interface configuration
  - Changes to running topologies
  - Console access
  - Image up/download
  - Pcap capturing (packet capturing)

- **graphical editor**
  - Easy to use
  - Full control over topology elements
  - Shows resource usage

- Configures network interfaces
  - IP addresses / Netmasks
Properties
- Bandwidth
- Latency
- Jitter
- Packet loss
- Corruption
- Duplication

On link bases:
Packet Capturing

- Properties
  - Captures packages on the wire
  - Direct filtering
  - Format: Pcap, (compatible with Wireshark)
  - Two modes
    - Download
    - Live capture
- Cloudshark
  - Online tool for packet analysis
- Packet capturing
  - On link bases
Executable Archives

- Archive contents
  - Software + dependencies
  - Start/Install script

- Archive actions
  - Upload: unpacks contents to a folder and runs start script
  - Download: packs folder into archive and transfers it to user
  - Status: displays the status of execution

- Use cases
  - Install software packages on VI
  - Run complete experiments
  - Upload/Download data

- Experiment lifecycle
  - Create executable archive
  - Upload archive, run experiment via start script
  - Download archive, contains results
Multiple VNC options
- HTML 5
- Java applet
- Client software
Application Areas

Access layer experiments
- Consider lower layers and hardware
  - Example: Mobile handover
- Requirements
  - Hardware access
  - Custom operating systems (Realtime)
  - Heterogeneous access technologies (3G, Wifi, etc.)
- Needs specialized testbeds depending on hardware NO
  - DES Testbed, Wisebed

Network layer experiments
- Focus on TCP/IP suite
  - Example: IPv6 extensions, TCP substitutes
- Requirements
  - Deep OS access (modified kernels, etc.)
  - Small but complex topologies, link emulation
- \textit{Tomato} offers
  - Full kernel access via KVM
  - Complex topologies
  - Link emulation
  - Packet capturing (for analysis)
  - Easy setup of topologies

Algorithm/Protocol experiments
- Work on top of network layer
  - Example: P2P-Networks
- Requirements
  - Huge but simple topologies
  - Link emulation
  - No hardware or OS access
- \textit{Tomato} offers
  - Lightweight virtualization with OpenVZ
  - Link emulation
  - Federation with other testbeds via Internet

Legacy software experiments
- Considers legacy software
  - „Legacy software” refers to any widespread software with undocumented or unpublished behavior
  - Example: Skype and Windows
- Requirements
  - Special environments, custom operating systems
  - Small but complex topologies
  - Link emulation and external packet capturing
- \textit{Tomato} offers
  - Custom operating systems with KVM (Windows
  - Access to external service via Internet connector
  - Packet capturing independent of guest OS

Paul Mueller, University of Kaiserslautern
Worldwide sites
Scaling up *Tomato*

- **Single-host deployment**
  - Hostmanager, Backend and Web-Frontend can run on the same host
  - Easy for local tests

- **Isolated multi-host setups**
  - Running multiple hosts with a single backend and web-frontend

- **Federated setups**
  - *Tomato* hosts can be used by multiple backends
  - The *Tomato* community consists of over 122 hosts at several sites

- **Testbed on demand**
  - Dynamically allocate cloud resources for experiments
  - Current research effort
    - Master thesis on allocating resources from CloudLab for *Tomato*
    - Bachelor thesis on dynamic allocation of resources
Use case 1: mobile devices

- **Scenario**
  - Services for agriculture *i*Green
  - Support for mobile devices
  - How does latency affect QoE and accuracy?

- **TOMaTo usage**
  - Simple topology
  - Special template with Android emulator
  - Usage of link emulation
Use case 2: Malware Analysis

- **Scenario**
  - Analysis of worm
  - Focus on network behavior
  - Fully contained topology

- **Topology**
  - Must run a Windows machine
  - External network (Internet) is connected to the victim to upload the virus
  - The external network is disconnected and the virus can be started
  - The links can be monitored to analyze the virus traffic

- **ToMaTo usage**
  - Simple topology
  - No connection to Internet
  - Usage of packet capturing
Use case 3: MAKI: Study of Transitions in Overlay-based Streaming Systems

Requirements for testbed experiments
- Scenario: 100+ machines, real and emulated network parameters
- Tooling:
  - Automatic, parametrized generation of ISP-like network topologies – custom script
  - Automatic deployment of software on testbed – custom script
- Planned: usage of OpenFlow-controlled software
  - Software switches: OpenVswitch
  - Hardware switches NEC

Steps taken
- Study of available features, first proof of concept (ToMaTo web frontend)
- Topology generation script (based on ToMaTo Python API)
Features
- Supports all actions that are available on WebUI
- Plus useful commands for programmatic work with topologies

Examples
- Creation of hosts
  ```python
  node = element_create(topId, nodeType, None, {"site": "ukl", "_pos": {"x":x,"y":y}})
  ```
- Creation of standard switches
  ```python
  switch = element_create(topId, "Inner Core",None,{"_pos": {"x":"0.5","y":"0.5"}})
  ```
- Running commands on hosts
  ```python
  element_action(node_id, "execute", {"cmd":"apt-get install rsync"})
  ```

Main lessons learned
- Run time consuming actions in parallel (especially prepare of elements)
- Use Linux (Windows works but is not tested well and requires changes to framework – Feature request including fixed were filed by us)
- Positioning of elements should be done in script (if use of WebUI is intendeded)
Current developments
- Extending and improving ToMaTo testbed
- Early work on scalable federated clouds
- Kaiserslautern has become an associated partner of the US CloudLab project

CloudLab
- New NSF funded project across the US
- Kaiserslautern has become an associated partner to the CloudLab project
- CloudLab offers bare metal machines
Tomato is a new network experimentation tool
- Based on virtualization (KVM, openVZ, …)
- Easy to use graphical front end
- Open-Source since version 2.0
- Ready to use: http://tomato-lab.org/

Feature rich environment
- For research and teaching

Sustainability
- Supported by Data Center at Kaiserslautern
- Part of new research projects (like MAKI)

Worldwide footprint
- Hopefully next time at ICSI/Berkeley

Testbed on demand based on bare metal cloud infrastructures
- Dynamic deployment of Tomato infrastructure to CloudLab and Chameleon
Questions?
Prof. Dr. Paul Mueller
Integrated Communication Systems ICSY
University of Kaiserslautern
Department of Computer Science
P.O. Box 3049
D-67653 Kaiserslautern

Phone: +49 (0)631 205-2263
Fax: +49 (0)631 205-30 56
Email: pmueller@informatik.uni-kl.de
Internet: http://www.icsy.de


Paul Müller, Dennis Schwerdel and Justin Cappos, *ToMaTo a Virtual Research Environment for Large Scale Distributed Systems Research*, PIK - Praxis der Informationsverarbeitung und Kommunikation, 2014.

Dennis Schwerdel, David Hock, Daniel Günther, Bernd Reuther, Paul Müller and Phuoc Tran-Gia, *ToMaTo - a network experimentation tool*, 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2011), Shanghai, China, April 2011.