Rethinking IO passthrough for virtual machines

<table>
<thead>
<tr>
<th></th>
<th>passthrough</th>
<th>mediation</th>
<th>emulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>userspace (using DPDK)</td>
<td>vpio, uio</td>
<td>vfio + vMux + vpio-user</td>
<td>vfio + OvSwitch + vhost-user</td>
</tr>
<tr>
<td>kernel</td>
<td></td>
<td>mdev, macVtap</td>
<td>vhost-net, vhost-blk</td>
</tr>
</tbody>
</table>

**General information**

**Advisor**  
Peter Okelmann

**Email**  
okelmann@in.tum.de
Virtual machines (VMs) are capable tools:
- Experimentation with software<->hardware protocols for hardware development
- Local development/testing environments for OS/low level development
- Security isolation in clouds

There exist plentiful ways to grant virtual machines access to physical devices:
- Passthrough: The entire PCI device is passed through to a single VM. Often, SR-IOV is used to split a single physical function (PF) into many virtual functions that can in turn be passed through.
- Mediation: Interactions between the driver and the software are mediated by vMux which can observe, pass through or modify all of them.
- Emulation: The host is running the device driver, a network stack and emulates a completely virtual device for the VM (VirtIO-net, e1000).

Goal of this thesis is to design and build a novel IO architecture that is more flexible, secure and performant than comparable existing solutions (vhost-user [1]) by making use of mediation. The value of mediation lies in allowing hybrid approaches, combining advantages of passthrough and mediation:
- Flexible:
  - Userspace device driver (-mediation/passthrough)
  - Unified control plane
  - Ease experimentation with protocols for hardware development
  - Ease experimentation with device mediation
- Secure: more code outside the hypervisor, containerize device emulation
- Performance: highest performance because full passthrough is still possible

### Keywords
C lang, Linux, VMs, SR-IOV / PCI virtual functions, device passthrough/mediation

### Goals

**Concrete outcomes**
1. Build a framework for experimentation (vMux [4], DPDK [1] + libvfio-user [2, 3])
2. Identify the core feature-set needed for high-performance operation of the Intel E810 NIC
3. Build a simple passthrough layer with vMux, comparable to virtual functions
4. Measure the system

**Bonus points**
5. Compare to virtual function performance
6. Device mediation to switch between emulation and passthrough
7. PCI support

### Prerequisites

**Compulsory**
- Experience in systems or driver programming
- Good knowledge of and experience with C.

**Preferred**
- Familiarity with DPDK, networking, kernel modules
- Python, Lua

### References
2. [https://github.com/nutanix/libvfio-user](https://github.com/nutanix/libvfio-user)
3. [https://sbrksb.github.io/2020/12/10/intro.html](https://sbrksb.github.io/2020/12/10/intro.html)
4. [https://github.com/vmuxIO/vmuxIO](https://github.com/vmuxIO/vmuxIO)

### Application process
Please send an email to the advisor including the following:
- Email subject: “Thesis application (DSE)”

Prof. Pramod Bhatotia
• CV
• A copy of your transcript(s)
• A short motivation statement, please include samples of your work that you are proud of (e.g., major projects, open-source contributions, Github page or similar, etc.) and/or writing samples (e.g., your technical blog, project reports, etc.)