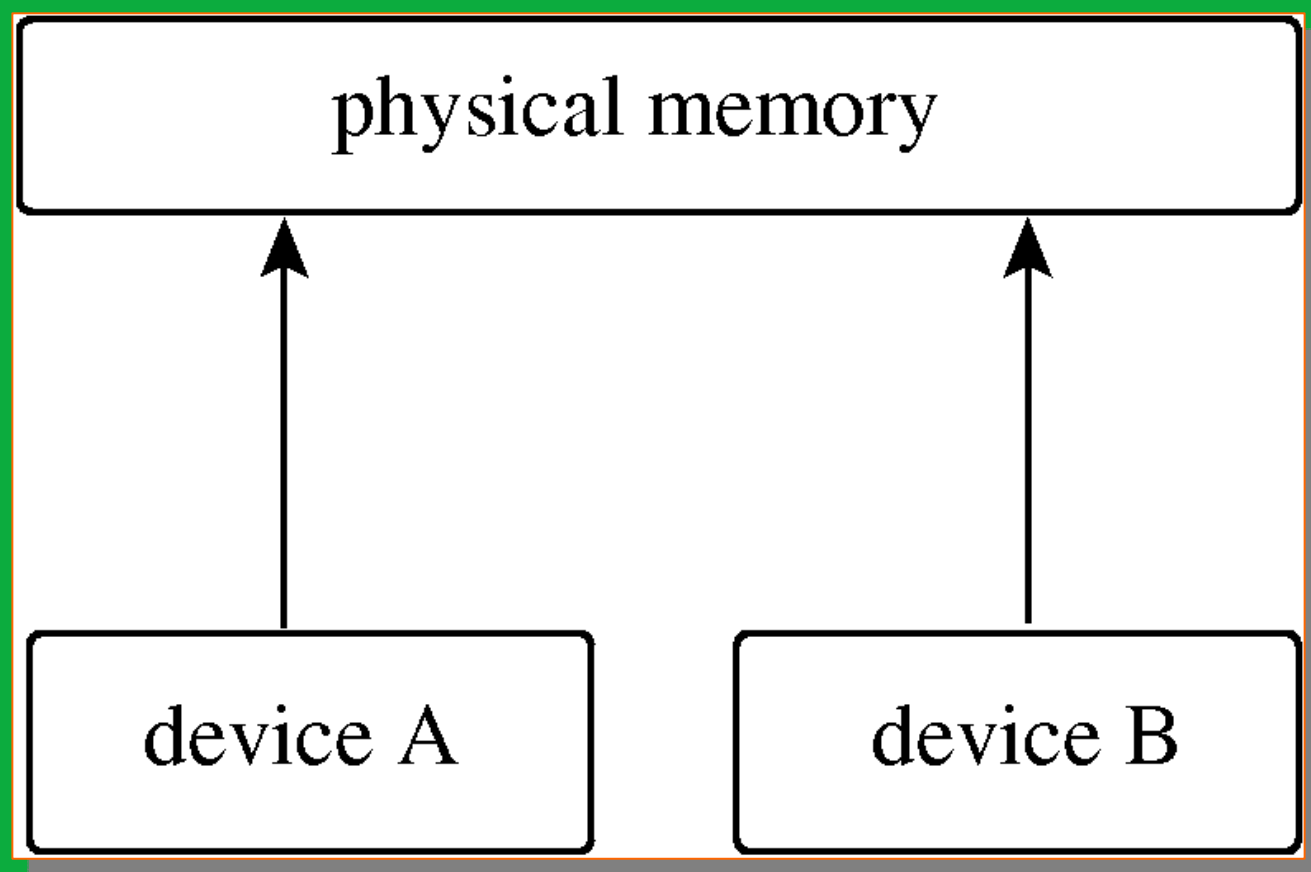


SCIMITAR: A Scalable I/O and Memory Pinning and Translation Architecture

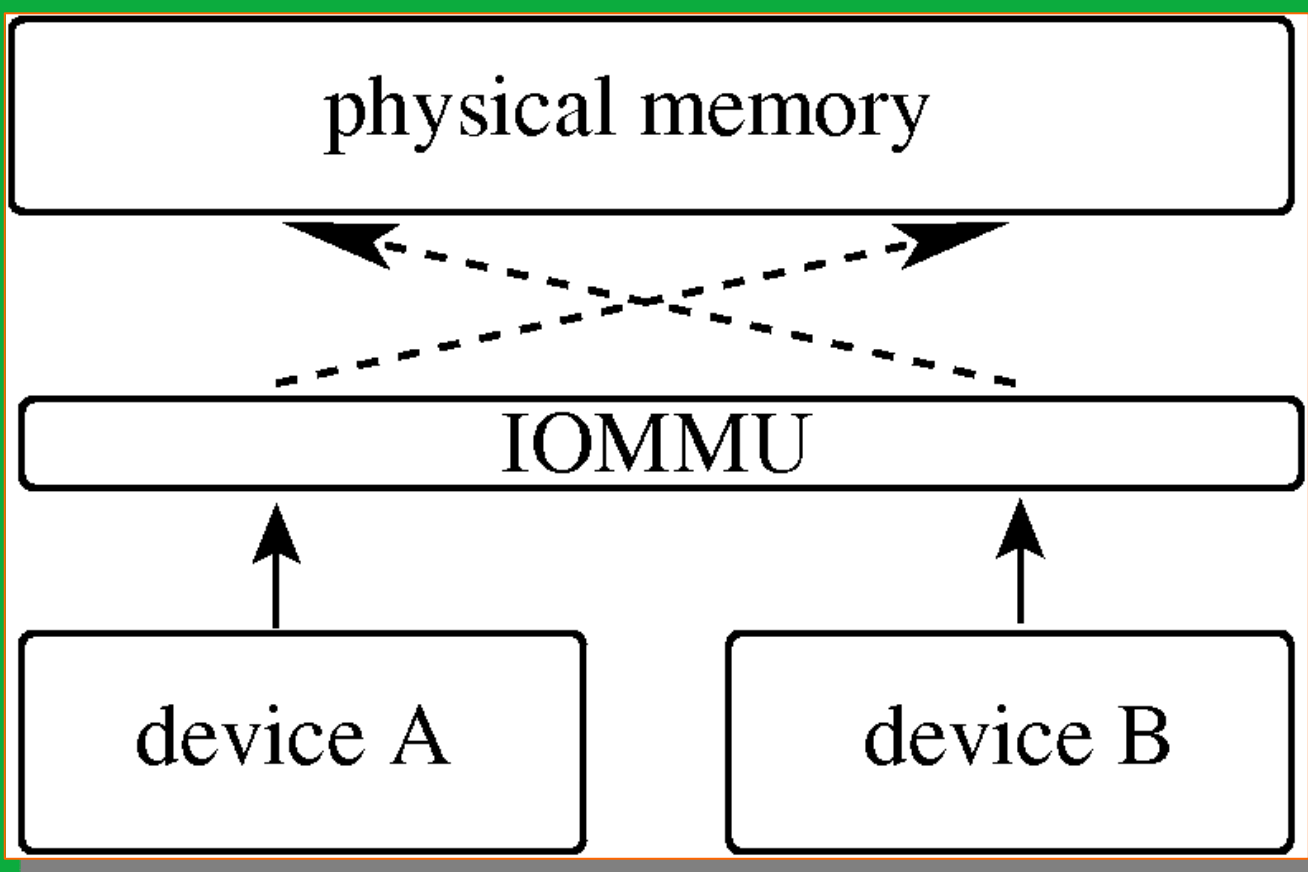
Muli Ben-Yehuda (muli@il.ibm.com),
Zorik Machulsky, Leah Shalev,
Julian Satran, Ben-Ami Yassour,
Orit Wasserman, Michael Vasiliev
(IBM Haifa Research Lab),
Yaron Weinsberg (Microsoft),
Emil Elazar (Hebrew U. in Jerusalem)

The SCIMITAR project explores virtual memory for I/O devices. Devices no longer share a single I/O address space, or have per-device I/O address spaces (IOMMU). Instead devices use consumer virtual addresses.

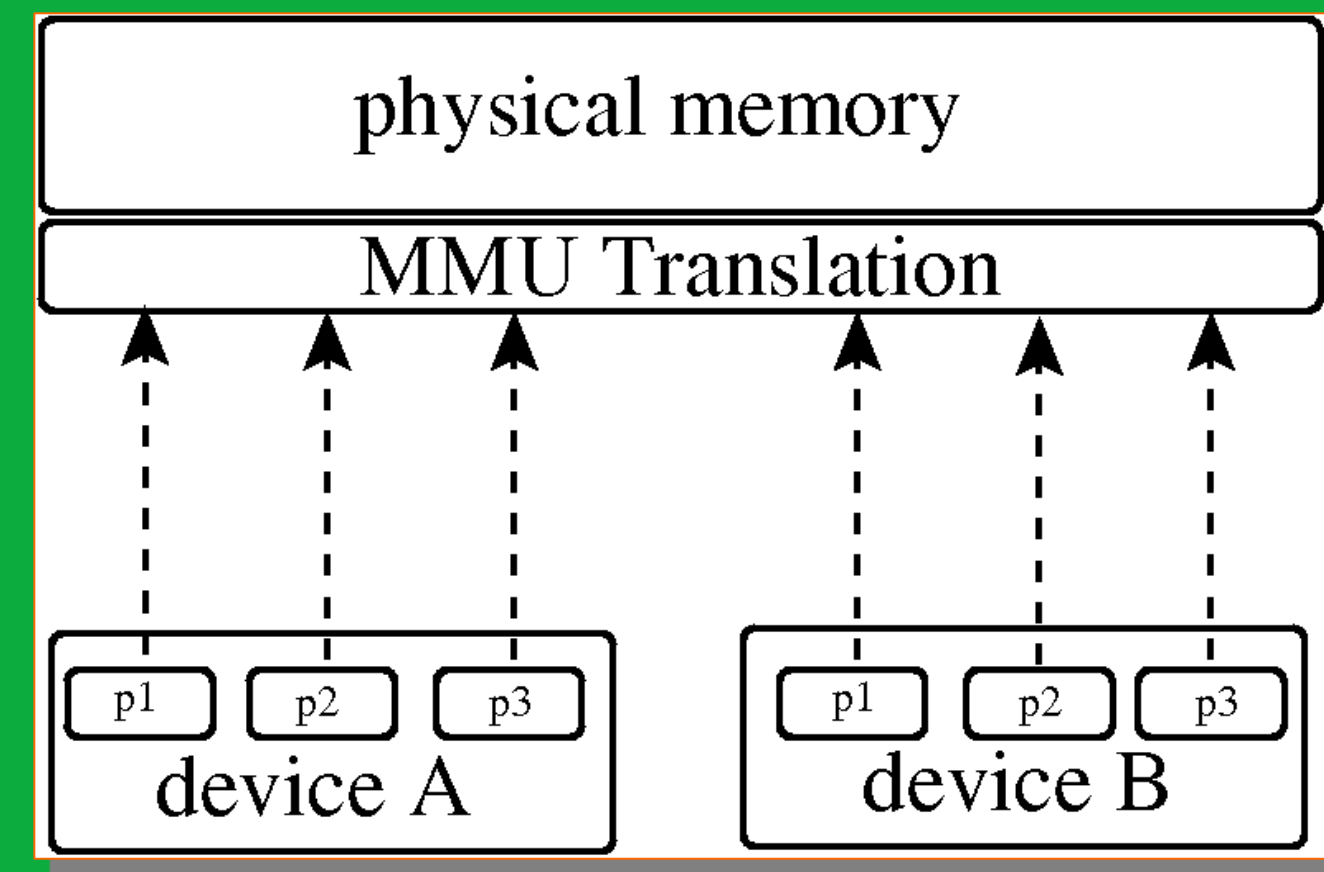
Single I/O address space



Per-device I/O address space



Consumer I/O address spaces



Motivation

- Memory pinning expensive & complex [Wyckoff et al., CCGRID '05]
- Direct access from virtual machines [Liu et al., USENIX '06, Ben-Yehuda et al., OLS '06]
- An I/O device is just another heterogenous core [Weinsberg et al., ASPLOS '08]



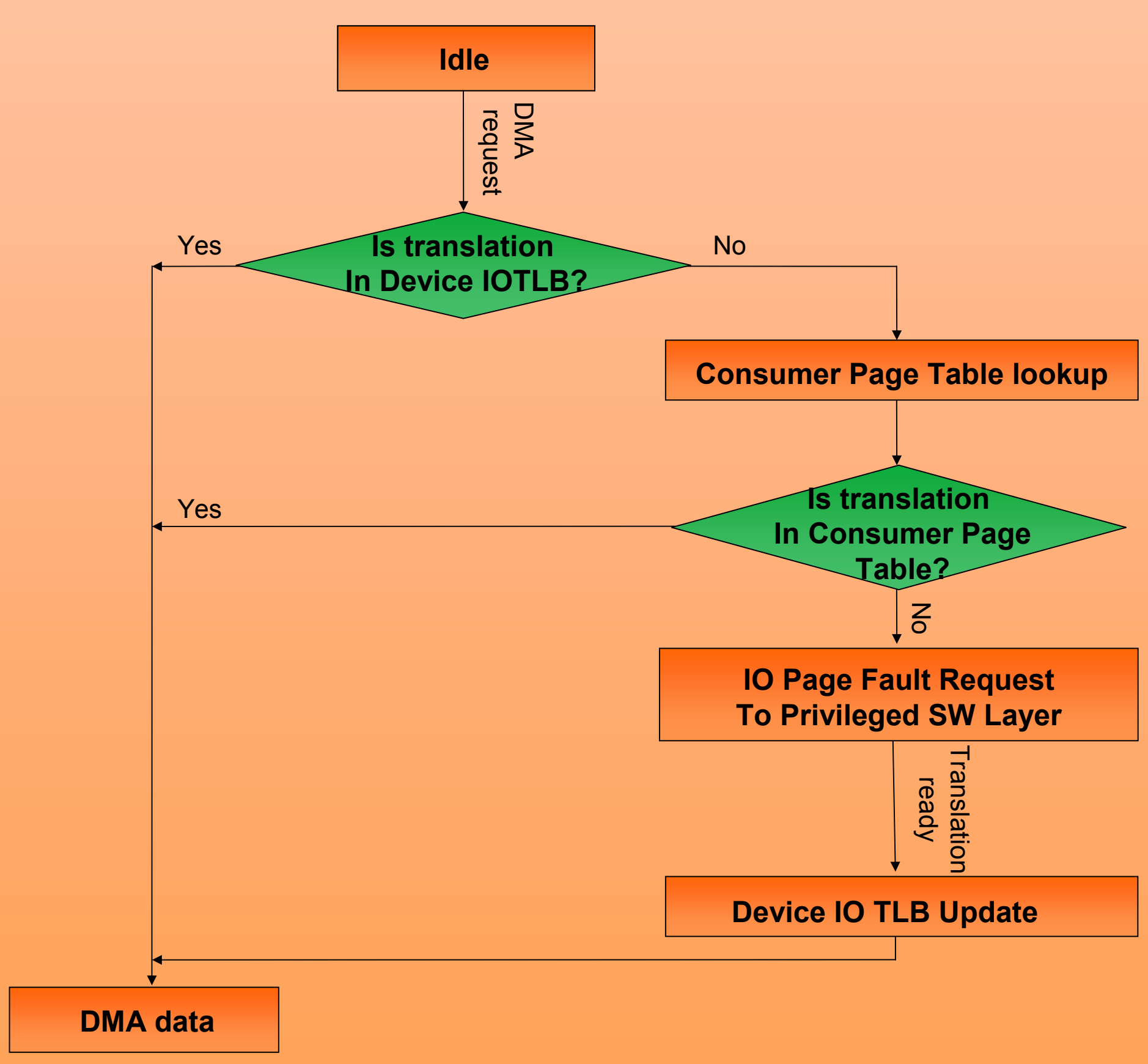
Assumptions and observations

- Memory frames which are DMA targets are usually already allocated and resident!
- Strategy: make the common case fast, and the unlikely case possible.

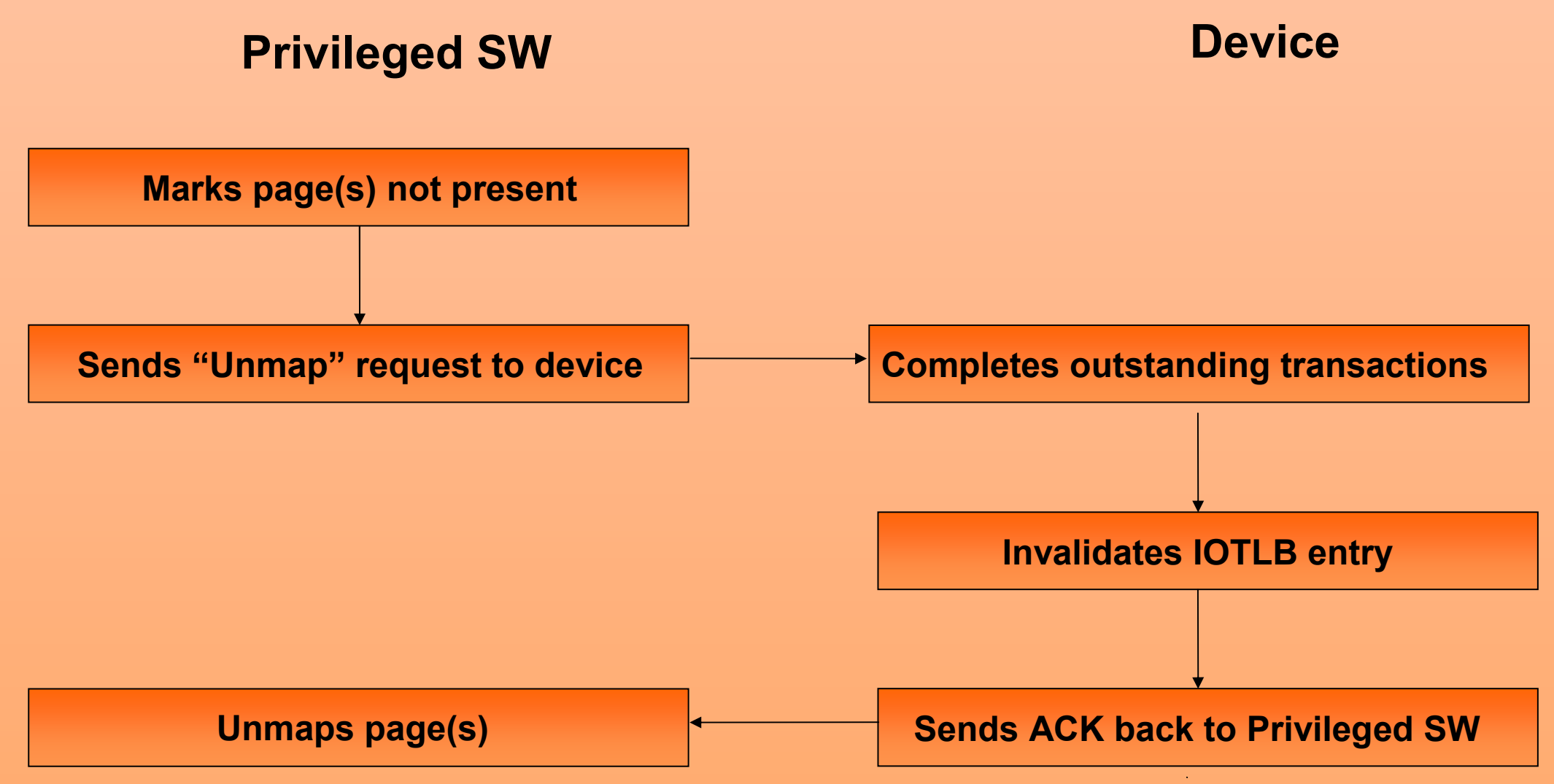
Theory of operation

- Use existing MMU translation mechanisms for virtual->physical translation.
- Don't pin up-front (assume memory is pinned!)
- Move pinning cost and complexity to rare unmap.

DMA Request Processing Flow Diagram



Memory Page Unmap Procedure



Experimental Prototype

- Using programmable network adapter.
- Rewrote firmware to raise I/O page faults and cache I/O translations (IOTLB).
- Working on operating system memory management changes (I/O page fault handler, unmap protocol).
- Evaluating underlying assumptions.

Some open questions

- How do you build such a system (HW and SW)?
- Does I/O have spatial and temporal locality?
- Are I/O page faults feasible?
- How do you build an optimal device IOTLB?

Related work

- Shrimp User-level DMA [Blumrich et al., HPCA '96]
- PCI-SIG IOV / ATS
- Quadrics, Infiniband, iWarp
- Pass-through with IOMMUs [Ben-Yehuda et al., OLS '06]