

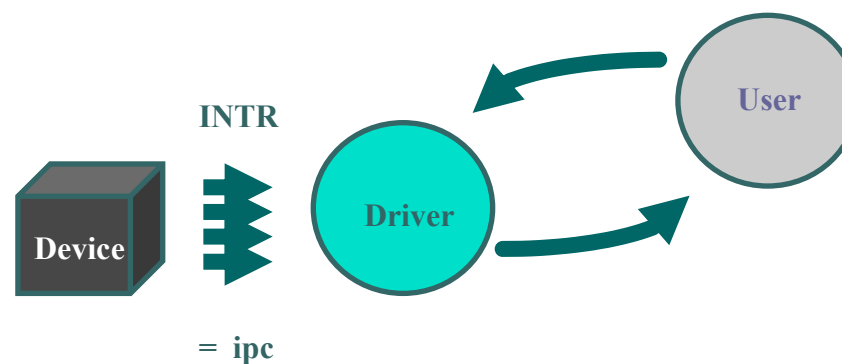
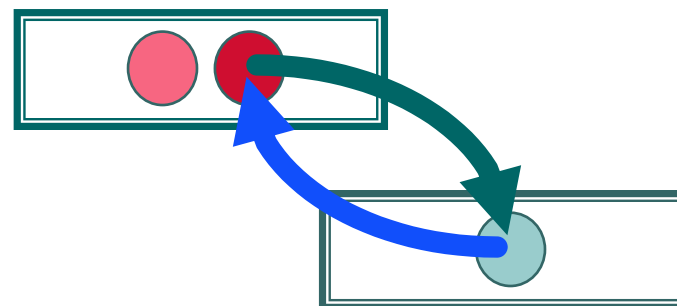


Future Directions in the Evolution of the L4 Microkernel

Kevin Elphinstone

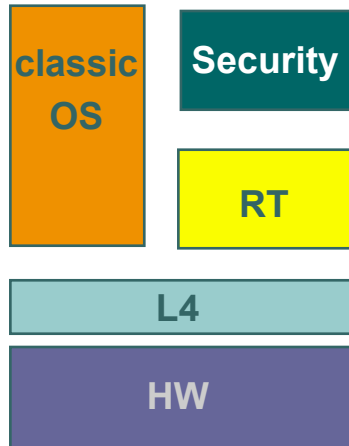
L4 Microkernel Introduction

- Few abstractions and mechanisms
 - Threads, address spaces
 - IPC, mapping
 - Traditional OS features implemented by servers
- Device drivers out of the kernel
 - Interrupts deliver via IPC
 - Device registers addressed via mapping
- Results in small kernel
 - Approx. 10K lines of code
 - 2/3 Platform independent

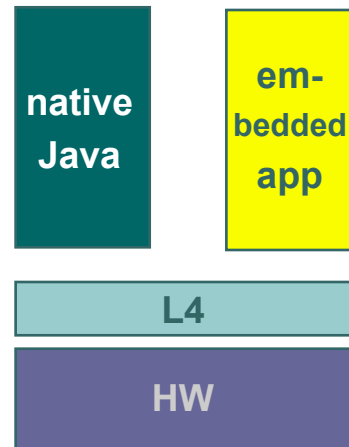


Example System Instances

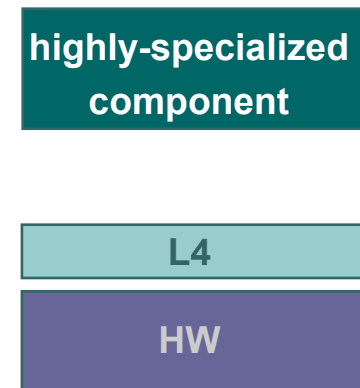
classic +



thin



specialized





L4 Goals

- *All things to all people*
 - A standard basis for system construction as a hardware architecture is now
 - Minimal policy, mostly mechanism
 - Good (in)direct control of the hardware
 - Thorough attention to performance related details
 - Bias towards “performance”



Intended Application Areas

Selection of topics from a DFG grant

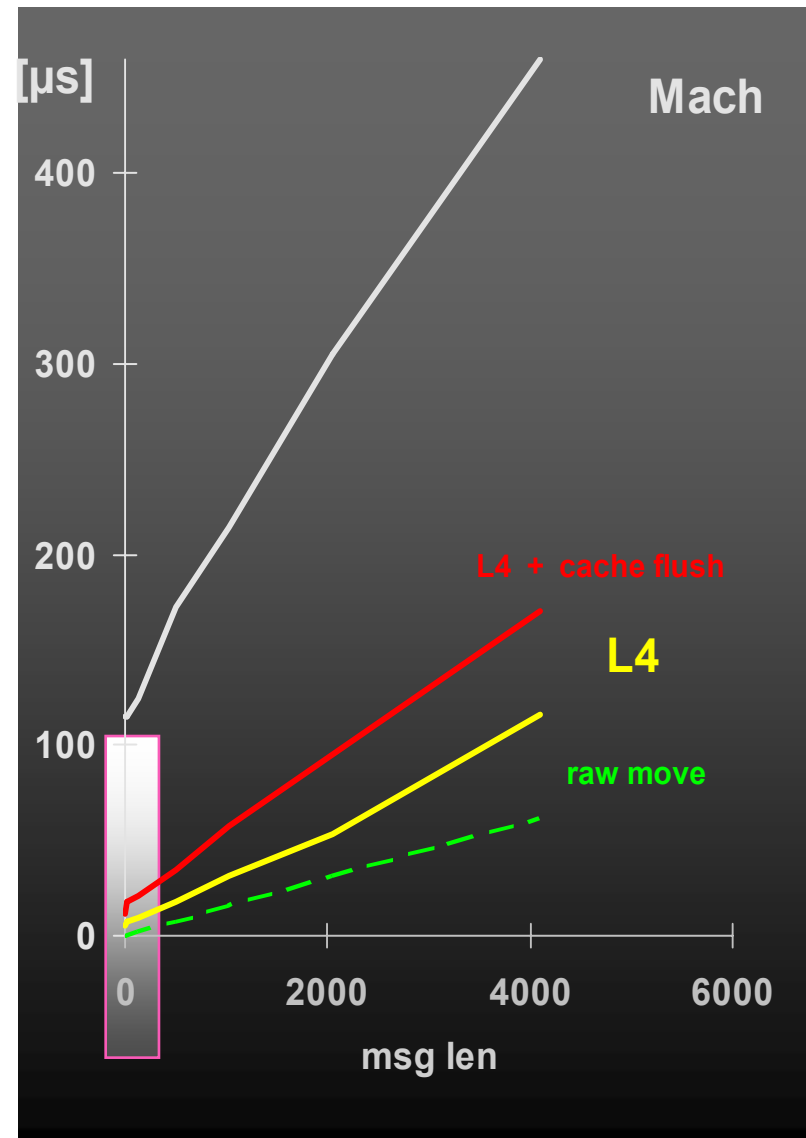
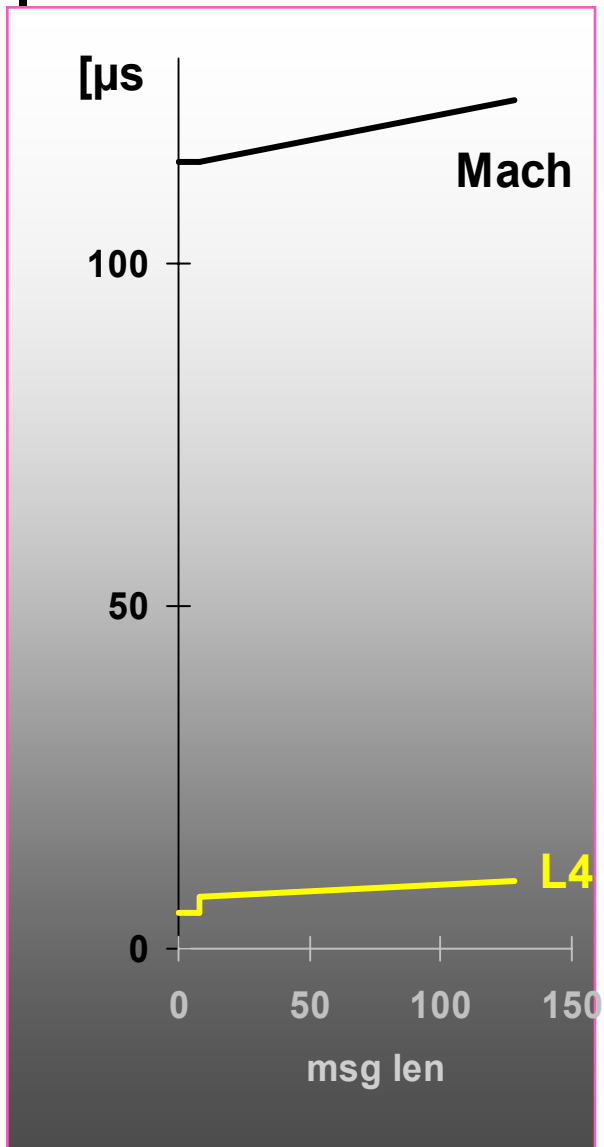
- Pervasive Computing
- Parallel and Cluster Architectures
- IDL compilers
- Power Control
- Multi-threaded Processors
- 64-bit Architectures
- L4/Linux
- SMP
- Real-time Systems
- Secure Systems
- Persistence
- SASOS

A decorative graphic consisting of three colored circles (dark teal, light teal, and grey) and a vertical line to their right.

OS Kernels

1. **Jochen Liedtke. Improving IPC by Kernel Design. In 14th Symp. on Operating Systems Principles, Asheville, North Carolina, December 1993. ACM.**
2. E. Skoglund, C. Ceelen, and J. Liedtke. Transparent orthogonal checkpointing through user-level pagers. In *Proc. 9th International Workshop on Persistent Object Systems (POS9)*, Lillehammer, Norway, September 2000. LNCS, Springer-Verlag.
3. Andreas Haeberlan and Kevin Elphinstone. User-level management of kernel memory. In *Advances in Computer System Architecture (Proc. ACSAC'03), Lecture Notes in Computer Science*, volume 2823. Springer-Verlag, October 2003.
4. H. Härtig, J. Wolter, and J. Liedtke. Flexible-sized page-objects. In *1996 Int'l Workshop on Object-Oriented in Operating Systems*. IEEE, 1996.
5. Jochen Liedtke and Horst Wenske. Lazy process switching. In *8th Workshop on Hot Topics in Operating Systems (HotOS)*, Schloss Elmau, May 2001.
6. Jochen Liedtke. Clans & chiefs. In *12. GI/ITG-Fachtagung Architektur von Rechensystemen*, Kiel, March 1992. Springer Verlag.
7. Jochen Liedtke. A Basis for Huge Fine-grain Address Spaces and User Level Mapping. In *Granularity of Objects in Distributed Systems, ECOOP workshop*, Kaiserslautern, July 1993.
8. **Jochen Liedtke. Toward real microkernels. *Communications of the ACM*, 39(9), September 1996.**
9. Jochen Liedtke. micro -kernels must and can be small. In *Proc. 5th Int'l Workshop on Object-Oriented in Operating Systems*, Seattle, WA, October 1996.
10. Jochen Liedtke, Kevin Elphinstone, Sebastian Schönberg, Hermann Härtig, Gernot Heiser, Nayeem Islam, and Trent Jaeger. Achieved IPC performance. In *6th Workshop on Hot Topics in Operating Systems (HotOS)*, Chatham, Massachusetts, May 1997.

IPC Costs (486, 50 MHz)



A decorative graphic on the left side of the slide consisting of three colored circles (dark blue, light blue, grey) and a vertical black line.

System Construction

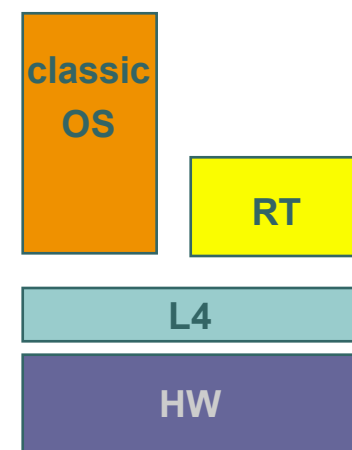
1. **Hermann Härtig, Michael Hohmuth, Jochen Liedtke, Sebastian Schönberg, and Jean Wolter. The performance of μ -kernel-based systems. In *Proc. 16th Symp. on Operating Systems Principles. ACM, 1997.***
2. Jochen Liedtke, Volkmar Uhlig, Kevin Elphinstone, Trent Jaeger, and Yoonho Park. How to schedule unlimited memory pinning of untrusted processes or provisional ideas about service-neutrality. In *HotOS*, March 1999.
3. **Jochen Liedtke, Vsevolod Panteleenko, Trent Jaeger, and Nayeem Islam. High-performance caching with the lava hit-server. In *USENIX Technical Conf., June 1998.***
4. Aron, Park, Jaeger, Liedtke, Elphinstone, and Deller. The sawmill framework for virtual memory diversity. In *ACSAC*, 2001.
5. Alain Gefflaut, Trent Jaeger, Yoonho Park, Jochen Liedtke, Kevin Elphinstone, Volkmar Uhlig, Jonathon Tidswell, Luke Deller, and Lars Reuther. The sawmill multiserver approach. In *SIGOPS European Workshop*, September 2000.
6. **Andreas Haeberlen, Jochen Liedtke, Yoonho Park, Lars Reuther, and Volkmar Uhlig. Stub-code performance is becoming important. In *Proc. USENIX/IEEE/ACM WIESS/OSDI*, October 2000.**
7. Jerry Vochtelloo, Kevin Elphinstone, Stephen Russell, and Gernot Heiser. Protection domain extensions in Mungi. In *Proc. 5th Int'l Workshop on Object Orientation in Operating Systems*, Seattle, WA, USA, October 1996.
8. K. Elphinstone, S. Russell, G. Heiser, and J. Liedtke. Supporting persistent object systems in a single address space. In *Proc. 7th Int'l Workshop on Persistent Object Systems*, Cape May, 1997.
9. K. Elphinstone, G. Heiser, and J. Liedtke. Page tables for 64-bit computer systems. In John Morris, editor, *Australasian Computer Architecture Conference*, Auckland New Zealand, January 1999. Springer Verlag, Singapore.
10. **Gernot Heiser, Kevin Elphinstone, Jerry Vochtelloo, Stephen Russell, and Jochen Liedtke. The Mungi single-address-space operating system. *Software Practice and Experience*, 28(9), July 1998.**
11. **Kevin Elphinstone and Stefan Götz. Initial evaluation of a user-level device driver framework. In P. Yew and J. Xue, editors, *Advances in Computer Systems Architecture, Proc. 9th Asia-Pacific Conference, ACSAC'04, Lecture Notes in Computer Science*, volume 3189, Beijing, China, September 2004.**



Real-time Systems

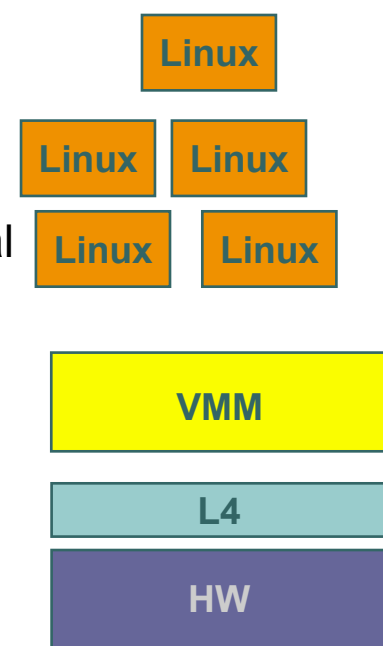
1. Michael Hohmuth and Hermann Härtig. Pragmatic nonblocking synchronization for real-time systems. In *USENIX Annual Technical Conference*, 2001.
2. Frank Mehnert, Michael Hohmuth, and Hermann Härtig. Cost and benefit of separate address spaces in real-time operating systems. In *Proc. 23rd Real-Time Systems Symposium*, December 2002.
3. H. Härtig, R. Baumgartl, M. Borriss, C. J. Hamann, M. Hohmuth, F. Mehnert, L. Reuther, S. Schoenberg, and J. Wolter. DROPS - OS support for distributed multimedia applications. In *Proc. 8th SIGOPS European Workshop*, Sintra, Portugal, 1998.
4. L. Reuther and M. Pohlack. Rotational-position-aware real-time disk scheduling using a dynamic active subset (DAS). In *Proc. 24th Real-Time Systems Symposium*, dec 2003.
5. Cl.-J. Hamann, J. Loeser, L. Reuther, S. Schönberg, J. Wolter, and H. Härtig. Quality assuring scheduling B₊ - deploying stochastic behavior to improve resource utilization. In *Proc 22nd Real-Time Systems Symposium*, December 2001.

classic +



Virtual Machine Monitors

1. V. Uhlig, J. LeVasseur, E. Skoglund, and U. Dannowski. Towards scalable multiprocessor virtual machines. In *3rd Virtual Machine Research & Technology Symposium (VM'04)*, May 2004.
2. J. LeVasseur, V. Uhlig, J. Stoess, and S. Götz. Unmodified device driver reuse and improved system dependability via virtual machines. In *Proc. 6th Symposium on Operating Systems Design and Implementations*, San Francisco, CA, December 2004.
3. J. LeVasseur and V. Uhlig. A sledgehammer approach to reuse of legacy device drivers. In *Proc. 11th SIGOPS European Workshop*, Leuven, Belgium, September 2004.
4. Jochen Liedtke, Nayeem Islam, Trent Jaeger, Vsevolod Panteleenko, and Yoonho Park. An unconventional proposal: Using the x86 architecture as the ubiquitous virtual standard architecture. In *SIGOPS European Workshop*, sep 1998.



A decorative graphic consisting of three colored circles (dark teal, light teal, and grey) and a vertical line to their right.

Security

1. Hermann Härtig. Security architectures revisited. In *Proc. 10th SIGOPS European Workshop*, Saint-Emilion, France, September 2002.
2. Jonathon Tidswell and Trent Jaeger. Integrated constraints and inheritance in dtac. In *RBAC Workshop*, July 2000.
3. Trent Jaeger and Jonathon Tidswell. An access control model to simplify constraint expression. In *ACM CCS*, November 2000.
4. Trent Jaeger, Jonathon Tidswell, Alain Gefflaut, Yoonho Park, Jochen Liedtke, and Kevin Elphinstone. Synchronous ipc over transparent monitors. In *SIGOPS European Workshop*, September 2000.
5. Trent Jaeger, Frederique Giraud, Nayeem Islam, and Jochen Liedtke. A role-based access control model for protection domain derivation and management. In *ACM RBAC Workshop*, nov 1997.
6. Trent Jaeger, Jochen Liedtke, and Nayeem Islam. Fine-grained protection in operating systems. In *USENIX Security Symposium*, January 1998.
7. Trent Jaeger, Jochen Liedtke, Vsevolod Panteleenko, Yoonho Park, and Nayeem Islam. Security architecture for component-based operating systems. In *SIGOPS European Workshop*, September 1998.
8. Trent Jaeger, Atul Prakash, Jochen Liedtke, and Nayeem Islam. Flexible control of downloaded executable content. In *ACM TISSEC*, May 1999.
9. Trent Jaeger, Kevin Elphinstone, Jochen Liedtke, Vsevolod Panteleenko, and Yoonho Park. Flexible access control using ipc redirection. In *HotOS*, sep 1999.



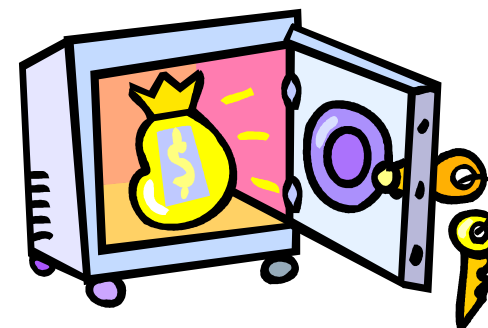
L4 a Success??

- 10+ years old
- 50+ papers directly or indirectly related to work involving L4
- 5+ Universities
- Some industry uptake
 - 4+ companies; most details under NDA
- ⇒ L4 is most things to some people
- Strengths and weaknesses largely influenced by stakeholder priorities

Future Direction: Trustworthy Embedded Systems

- “Trustworthy” in the sense of confidence in operation
 - Not “trusted” in TCB terminology, i.e. has the ability to violate security policy
- MMU-based systems
 - PDAs, phones, set-top boxes, home gateways/firewalls, etc....
- Selected application areas
 - Secure Systems
 - Enforcing a security policy
 - Dependable Systems
 - Justifiable reliance
 - Digital rights management
- Assurance

“Secure” Systems



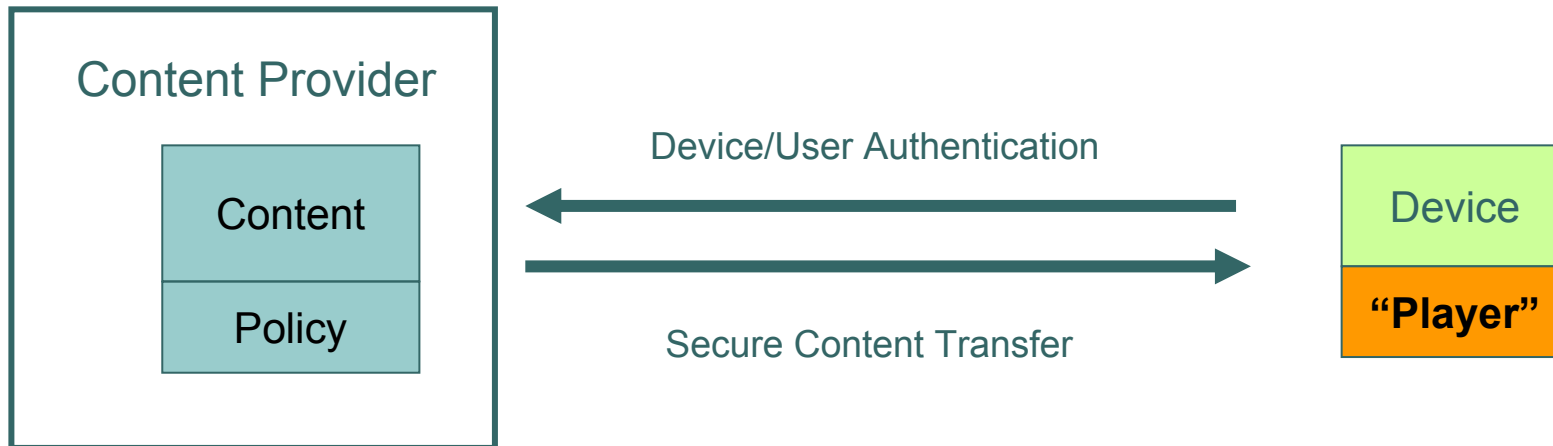
- The usual suspects
 - Confidentiality, Integrity, Availability
- The usual problems
 - Covert channels
 - Availability (Quality-of-service, Real-time)
 - Additionally: potentially hostile environments/invasive attacks are an issue in embedded system
 - Hardware tamper resistance
- Goal is to provide the basic mechanisms to enforce security policy
- Interesting Aside: What are sensible security policies in embedded devices?

Dependable Systems

- Approaches: Fault tolerance, fault masking, redundant hardware, n-version programming
- Fault isolation \Rightarrow partitioning
 - Ensure failures independent
 - No follow-on failures
 - Improve tractability of “system” analysis
 - Spatial partitioning
 - Software cannot change the private data of another partition
 - Achievable via access control
 - Temporal partitioning
 - Service received from shared resource is not influenced by another partition
 - Resources: memory, CPU, network bandwidth
 - Sources of covert channels violate temporal partitioning
 - Even though covert channels don't matter
 - Adding noise does not help

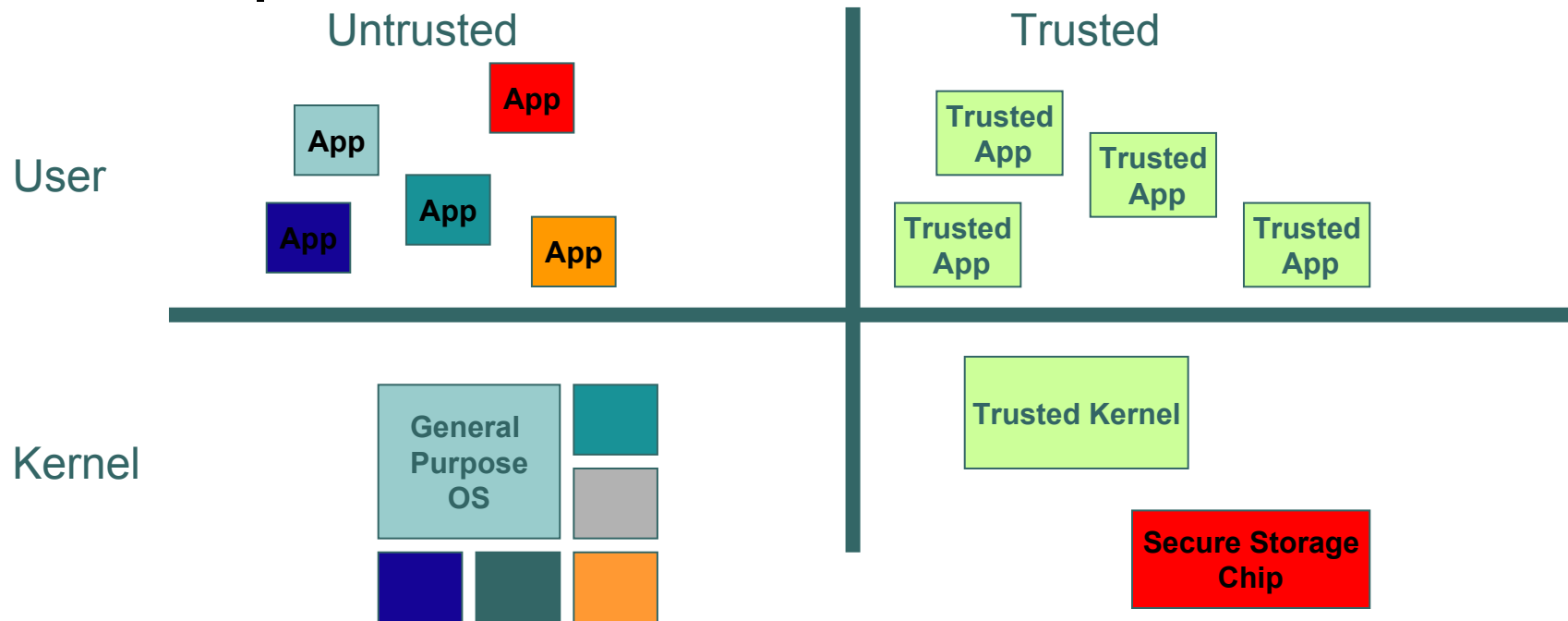


Digital Rights Management



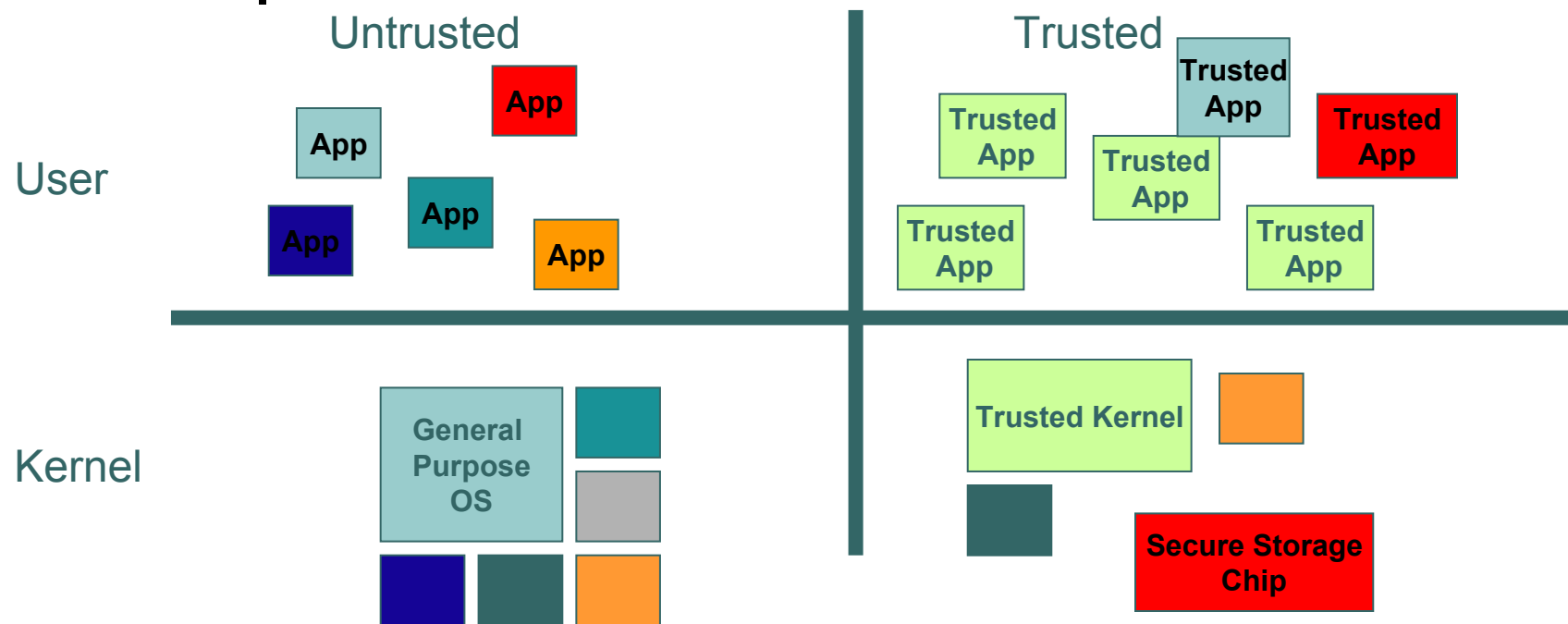
- What happens in the player?
- Content provider wants assurance that the “player” respects his usage policy
 - Easier for “enclosed” systems (e.g. DVD players, Satellite set-top boxes)
 - Tamper-resistant hardware implementations

DRM General Direction



- New trusted processor mode
- Secure Storage Chip provides
 - Hardware protected secret
 - Attestation of trusted kernel

DRM General Direction



- Avoid the right side looking like the left
 - “Trusted” is a label not a property
- Need take different approach
 - Small kernel, fine-grain protected components
 - L4, EROS

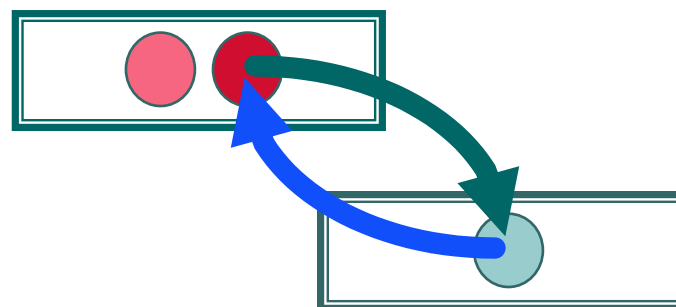


Selected L4 Issues

- Communication Control
- Kernel Resource Management

Communication Control

- Task interaction via IPC
 - ⇒ Control of information flow and interaction requires control of IPC
- Various models proposed and used, none of which “stuck” satisfactorily
 - Clan and Chiefs
 - Too many IPCs in general case
 - Redirection
 - Open questions on deception (forwarding) costs
- Looking at capability based approaches
 - Local namespaces
 - Fine grain control
 - Efficient (See EROS)



A decorative graphic consisting of three colored circles (dark teal, light teal, grey) and a vertical line to their right.

Kernel Resource Management

- Some Kernel Operations result in implicit storage allocation
 - Example: Thread control blocks during create thread
 - Currently central memory pool ⇒ possible DOS
- Exploring Black, White, and Grey options
 - Black = User-level controllable per task memory pools
 - Implicit allocation in kernel (Black box)
 - Issue: Cases where differentiation desirable
 - White = all kernel objects are first class
 - Issue: Hardware defined data structures and granularity of protection
 - Fine grain objects result in duplicate book keeping
 - Various Grey options
 - First class objects where there is value-add
 - Distinguish between classes of implicit objects
- Exploring options requiring less/no implicit data allocation.
- Not that many internal dynamically allocated data types



Final Comments

- Desirable properties for a trustworthy embedded systems kernel
 - Small
 - Simple and well understood
 - Efficient
 - Flexible
 - Assurance
- Goal: Future L4 will fulfil that role.