The Next Frontier of Cloud Computing is in the Clouds, Literally

Silviu Craciunas, Andreas Haas
Christoph Kirsch, Hannes Payer
Harald Röck, Andreas Rottmann
Ana Sokolova, Rainer Trummer

Joshua Love
Raja Sengupta

Universität Salzburg

UC Berkeley

Google Tech Talk, Mountain View, September 2010

The JAviator

javiator.cs.uni-salzburg.at

Quad-Rotor Helicopter

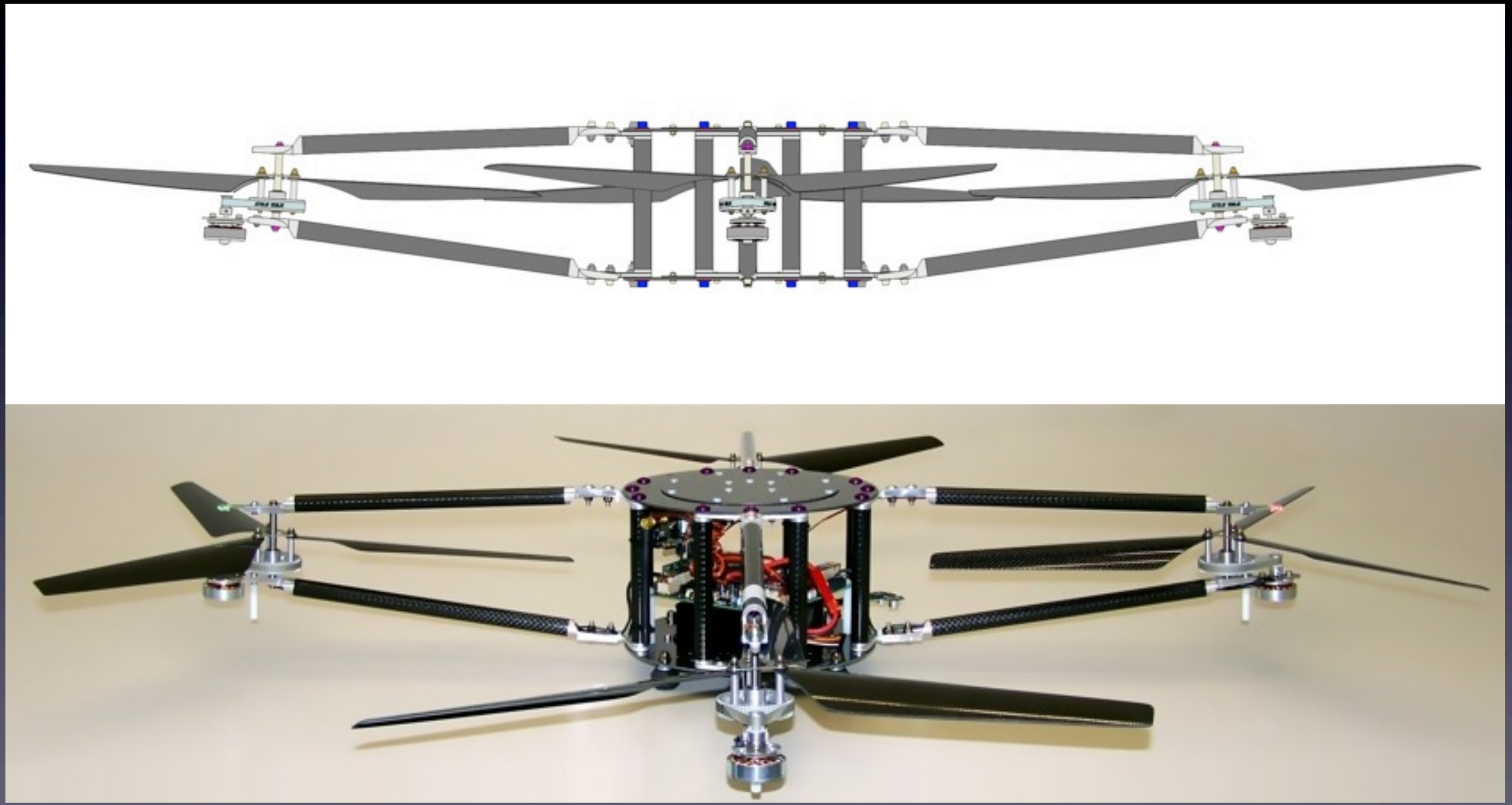
[AIAA GNC 2008]

- all carbon, titanium, aluminum design
- custom motors

- 1.3m diameter
- ~2.2kg weight
- +2kg payload

- ~40min (empty)
- ~10min (full)

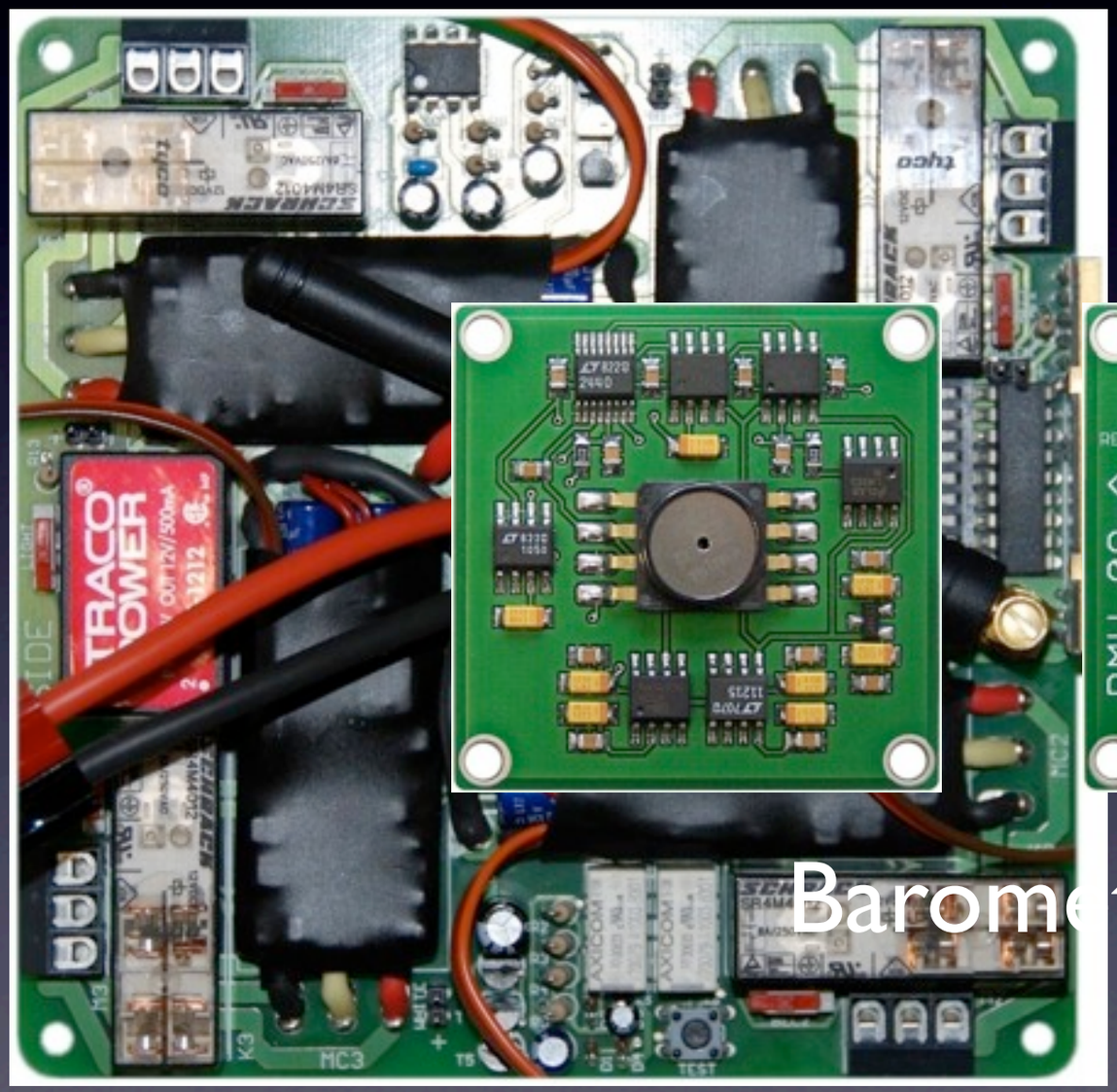
Open Source Blueprints



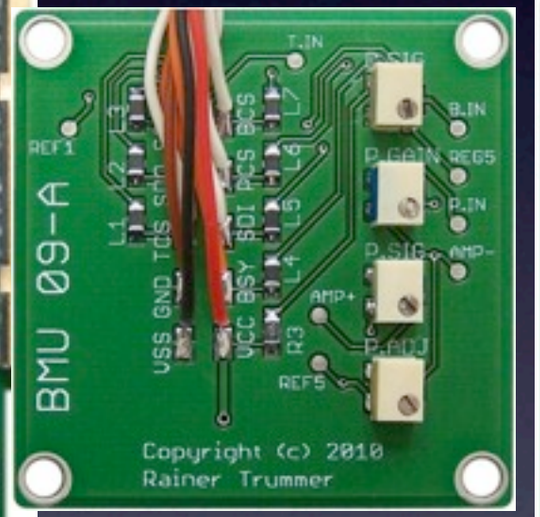
Minimal # of Different Parts

15V
10A
35g
1Kg

Custom Electronics



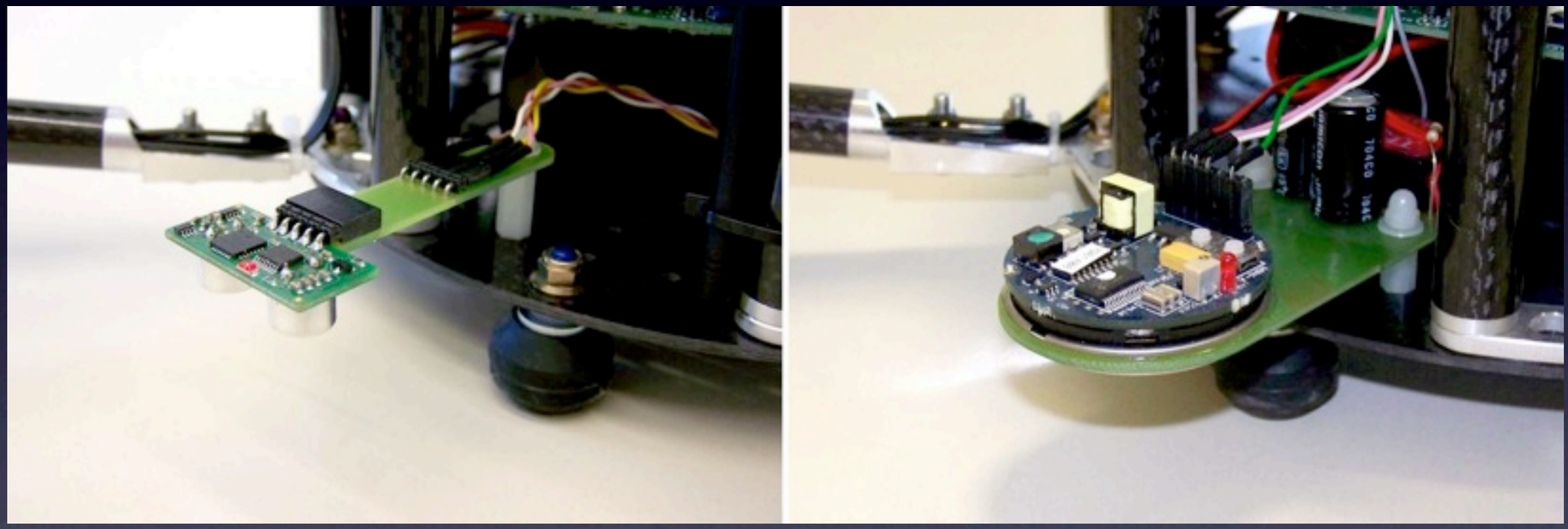
Power



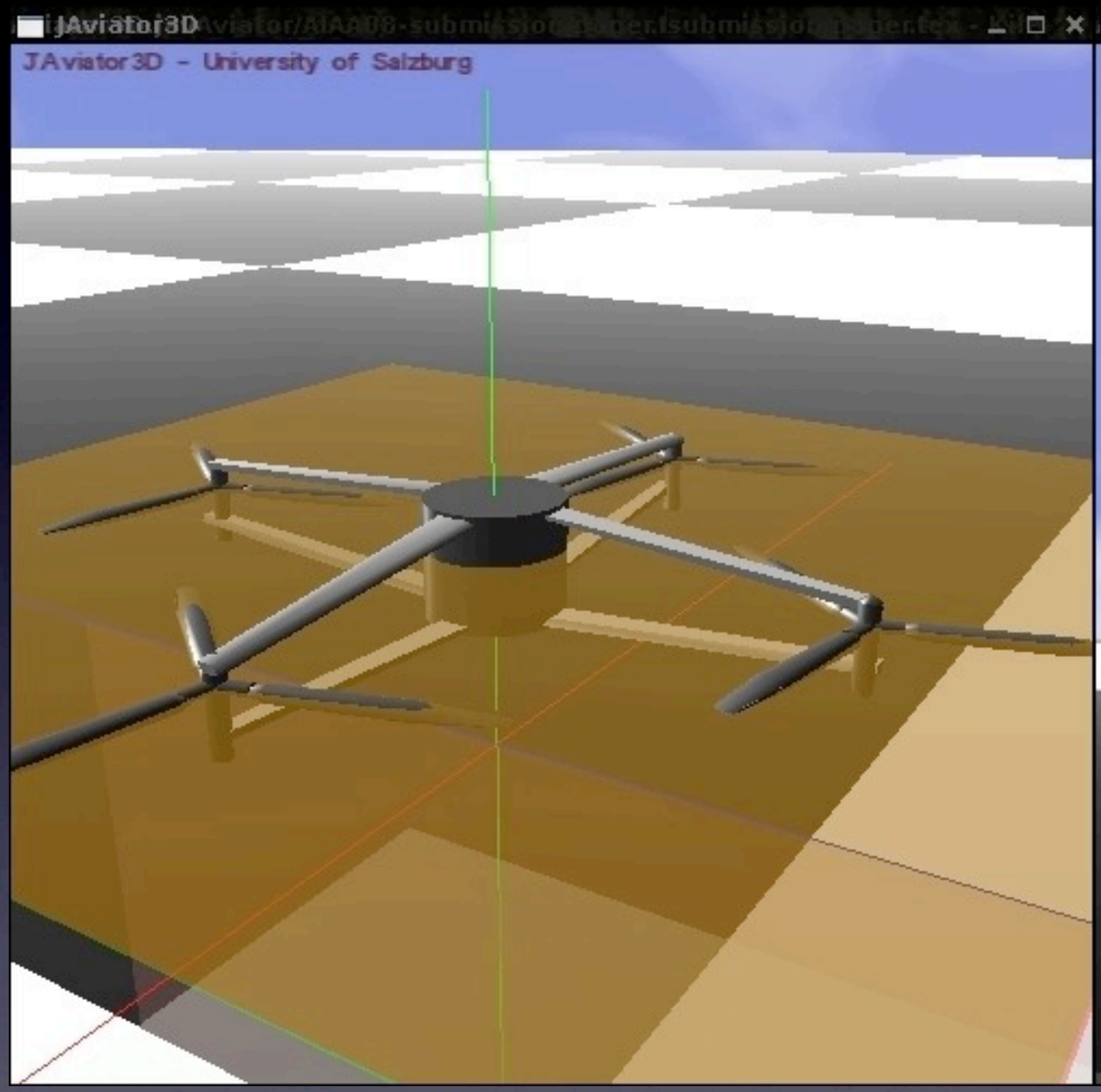
Barometer

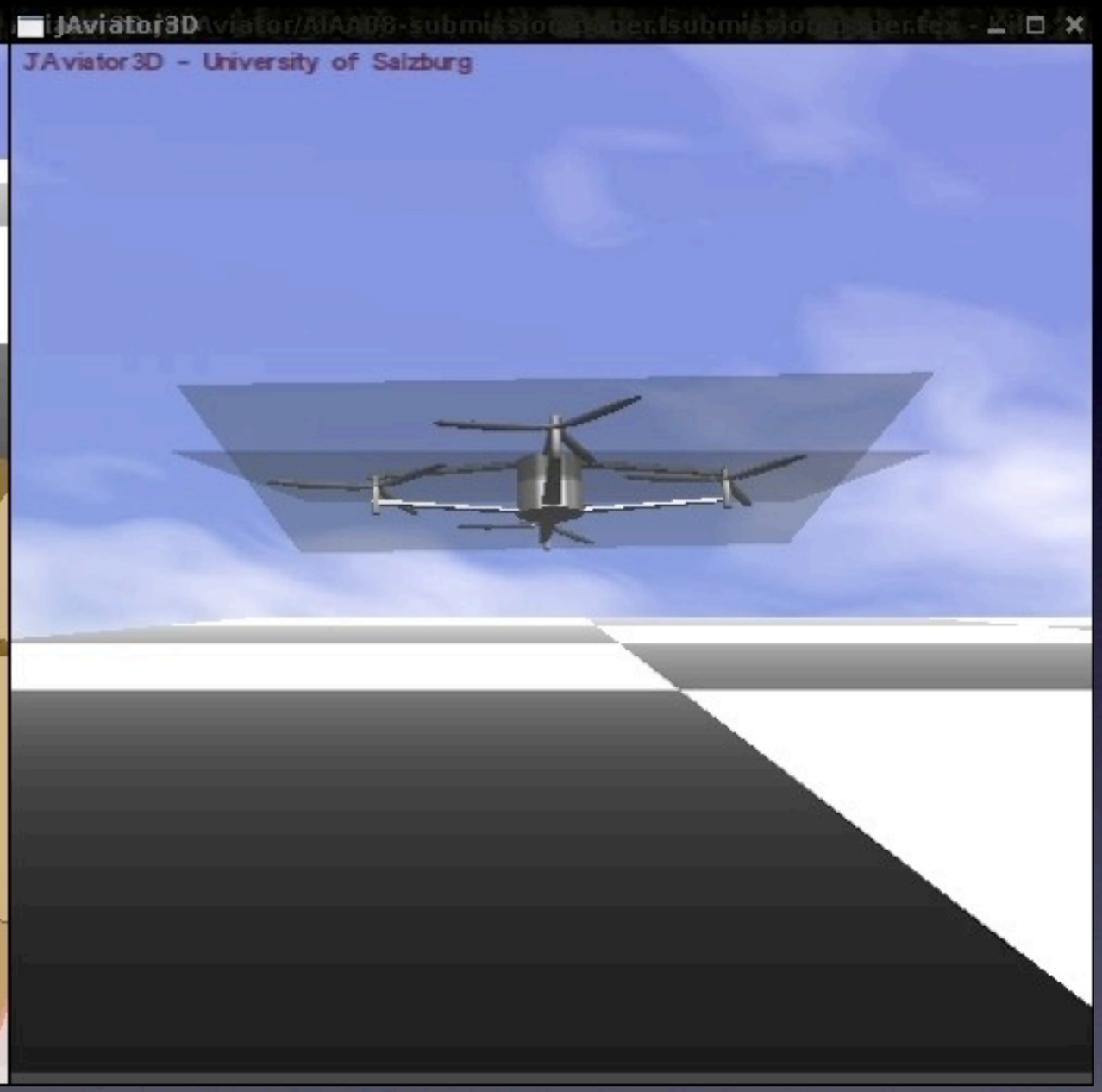
Remote

Off-the-Shelf Stuff

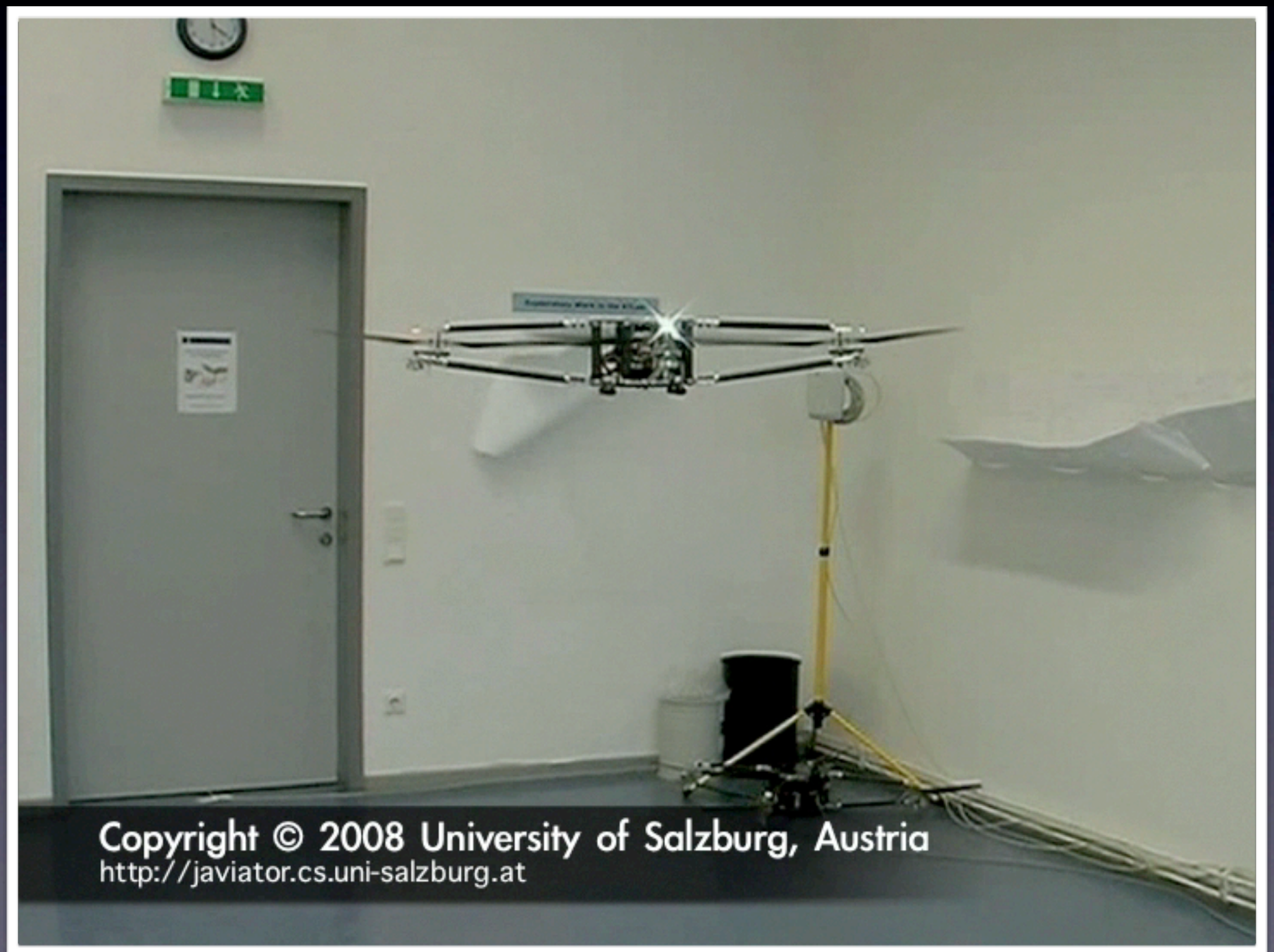


UWB RFID Ultrasonic Laser





Indoor Flight
STARMAC Controller

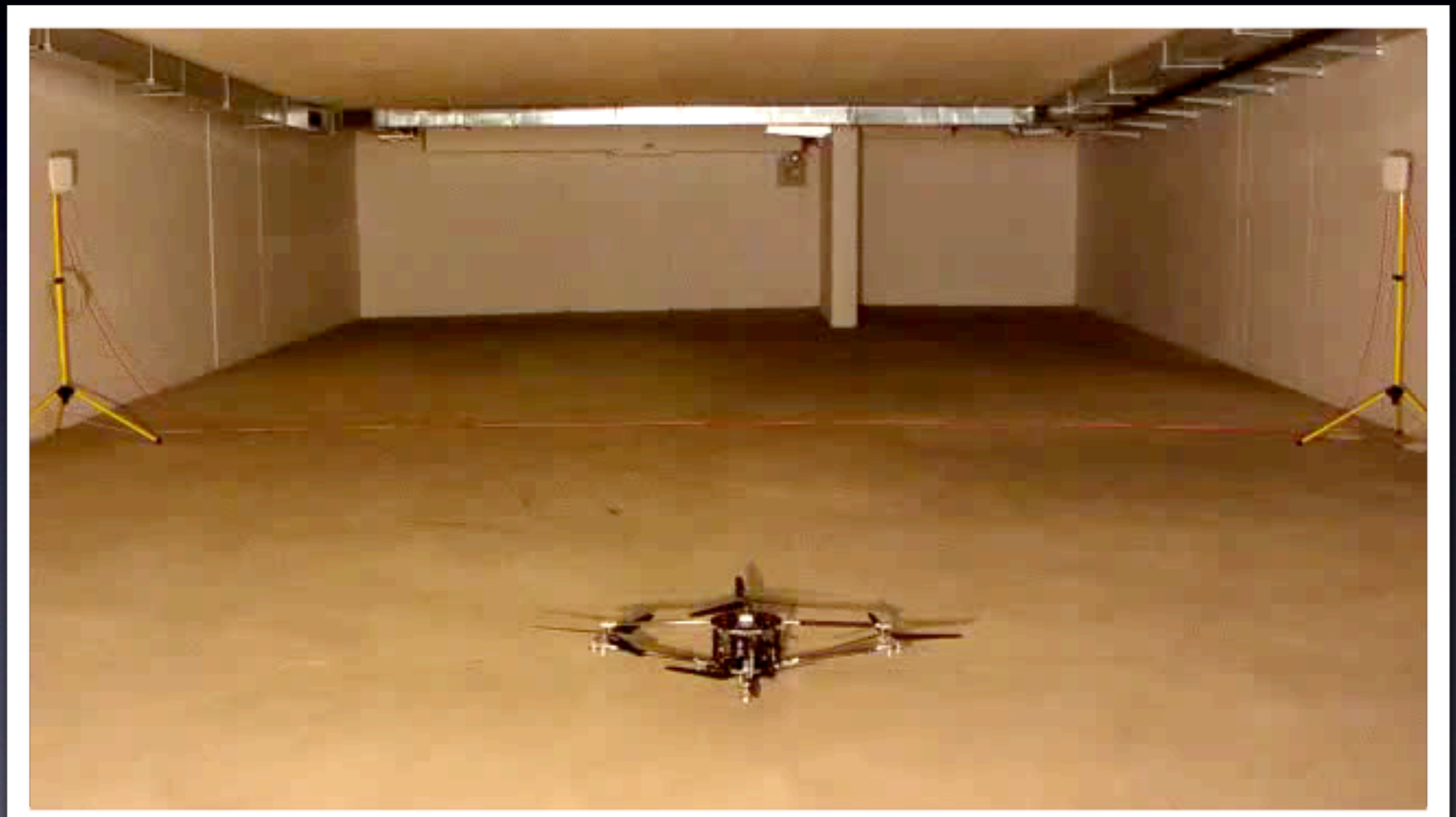


Outdoor Flight Salzburg Controller

More Recent: Yawing

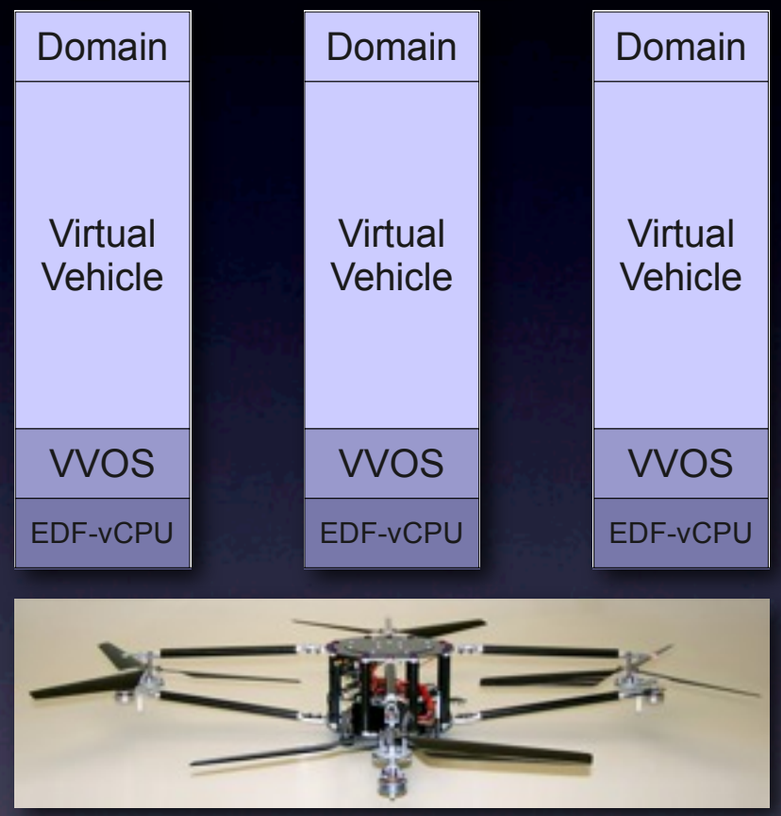
Oops

Autonomous



A Cyber-Physical Server

- IP address
- location
- capabilities
- motion

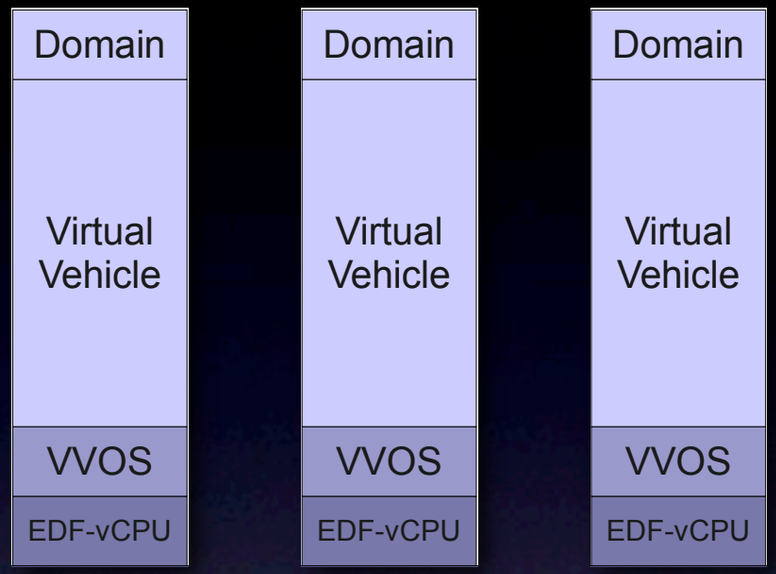


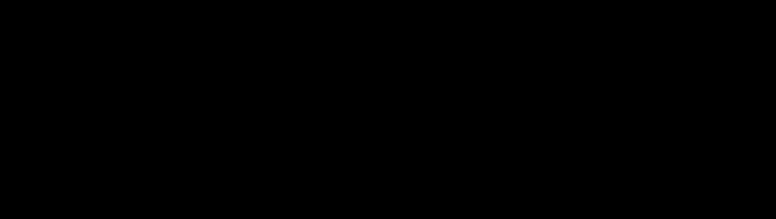
restricted

- IP address
- location
- capabilities
- motion

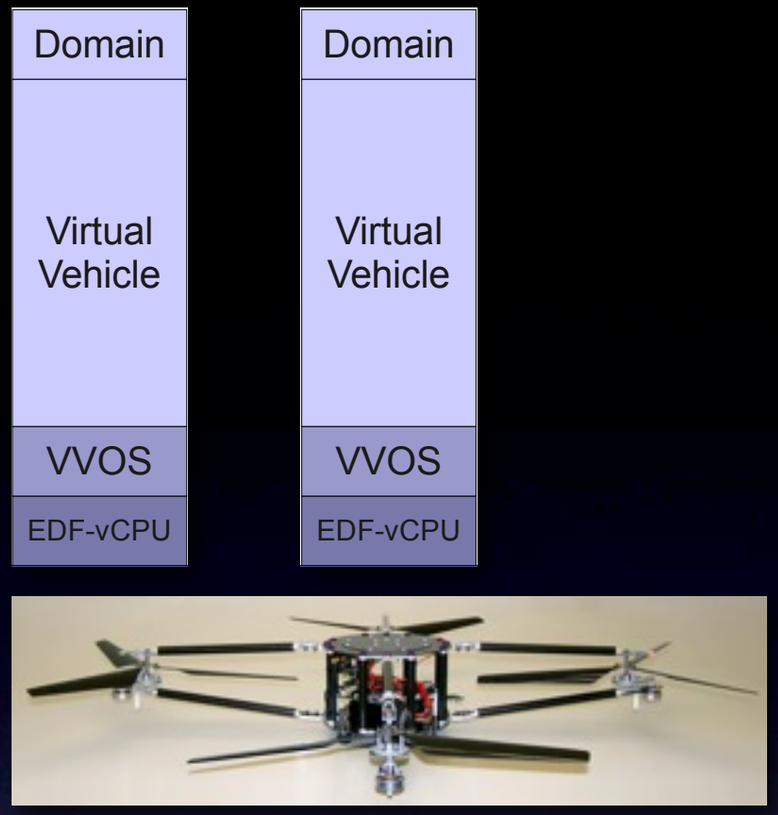
idealized

- IP address
- location
- capabilities
- motion

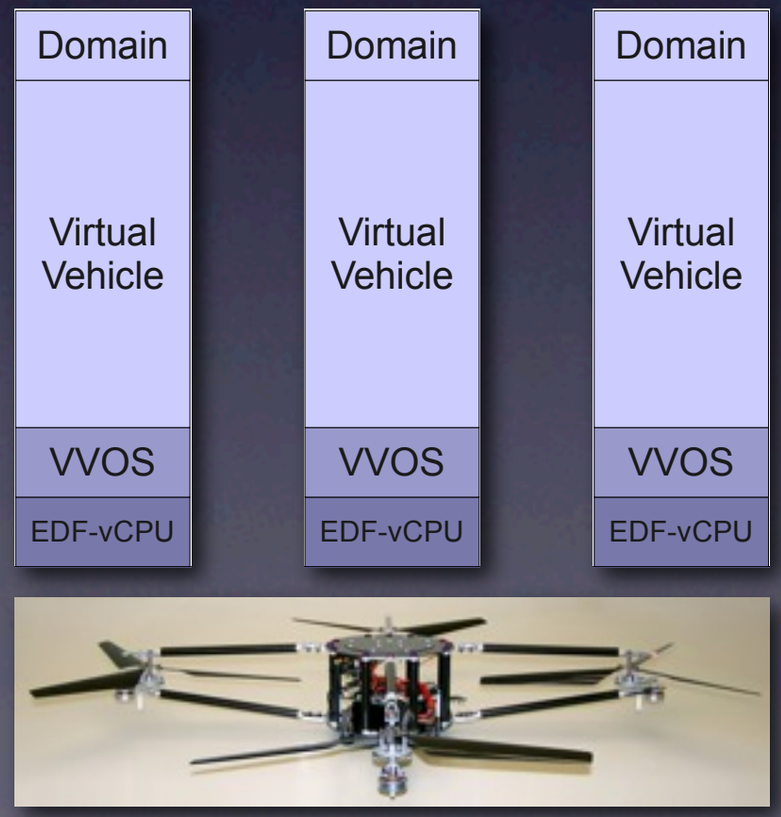




A Cyber-Physical Cloud [HotCloud 2010]



migration
=
flying



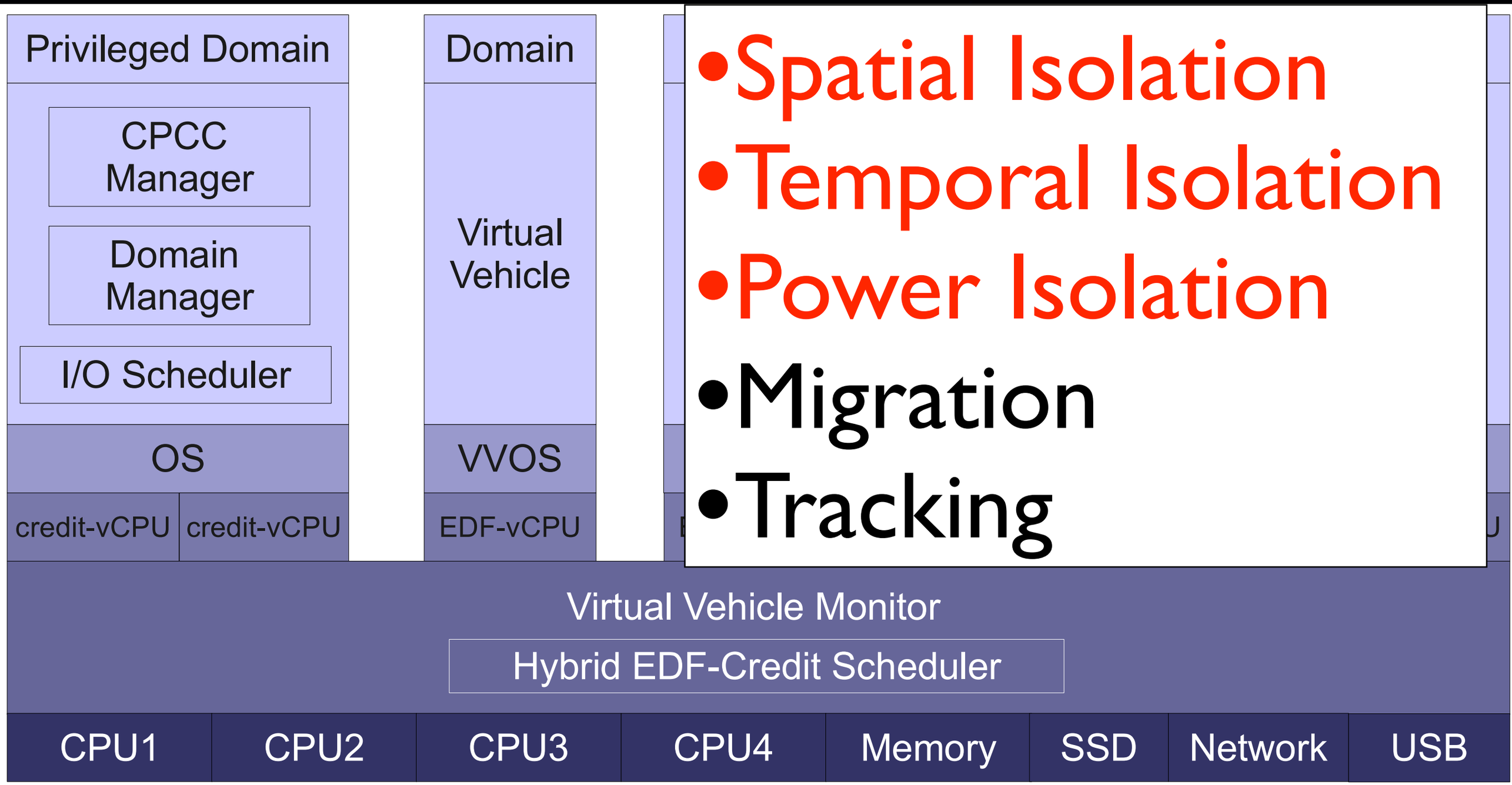
Goals

- Multi-provider (10s):
  - heterogeneous operations
- Multi-vehicle (100s):
  - heterogeneous systems
- Multi-task (1000s):
  - heterogeneous missions

High-Level Challenges

- Virtualization Infrastructure
  - Salzburg
- Collaborative Control
  - Berkeley
- Programming Language
  - Berkeley, Salzburg

Virtualization Infrastructure



- Spatial Isolation
- Temporal Isolation
- Power Isolation
- Migration
- Tracking

Isolating
space, time, power
simultaneously
likely requires
adequate runtime support
but also
advanced program analysis

We need
runtime environments
with an
interface
to
program analysis
for
trading off complexity

Heap Management

Do We Need Compaction?

- Compact-fit explicit heap management [USENIX ATC 2008]:
  - malloc and free are constant-time, unless compaction is necessary
    - memory is kept size-class compact
  - fragmentation is history-independent and predictable in constant-time
  - partial compaction: program analysis!
- C code available at:
  - tiptoe.cs.uni-salzburg.at/compact-fit

Probably not

And Garbage Collectors?

- Short-term memory for self
  [SBG10, submitted]:
  - all operations are const
  - constant per-object spac
- Java patch under EPL
  - based on Jikes RVM, GN Classpath class library
- Dynamic C library (libscm) under GPL
  - based on POSIX threads, ptmalloc2 allocator
- Available at:
  - tiptoe.cs.uni-salzburg.at/short-term-memory

works with any legacy code (1-word space overhead per memory block)

Hopeful!

Short-term Memory

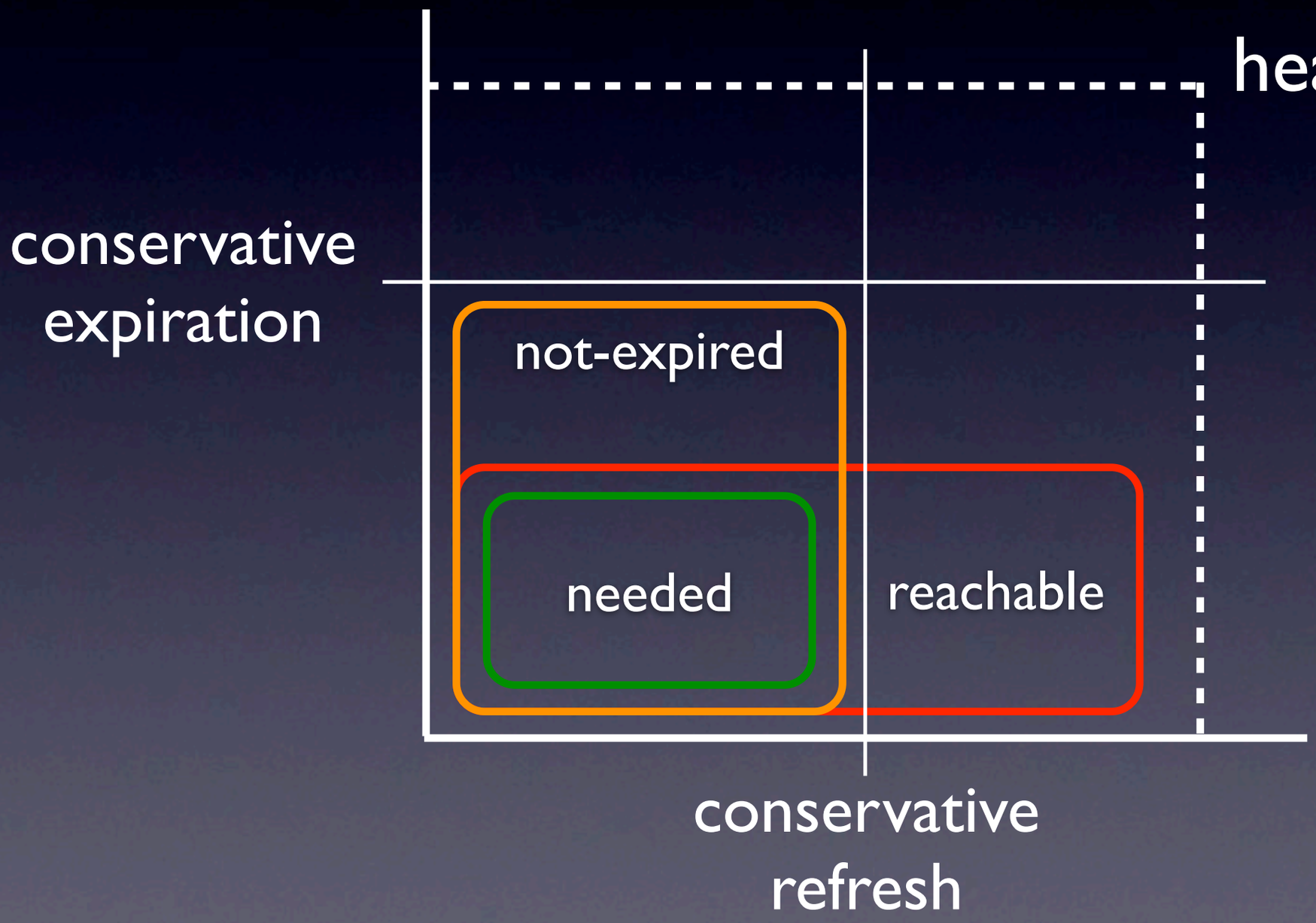
- Next week, Tue, Sept 7, 4pm @ UC Berkeley
- Memory objects are only guaranteed to exist for a finite amount of time
- Memory objects are allocated with a given expiration date
- Memory objects are neither explicitly nor implicitly deallocated but may be refreshed to extend their expiration date

With short-term memory
programmers or algorithms
specify which memory objects
are still needed
and not
which memory objects are
not needed anymore!

Explicit Programming Model

- Each thread advances a thread-local clock by invoking an explicit `tick()` call
- Each object receives upon its allocation an expiration date that is initialized to the thread-local time
- An explicit `refresh(`*Object*`,` *Extension*`)` call sets the expiration date of the *Object* to the current thread-local time plus the given *Extension*

Heap Management



Sources of Errors:

1. not-needed objects are continuously refreshed or time does not advance (memory leaks)
2. needed objects expire (dangling pointers)

Our Conjecture:

It is easier to say
which objects are still needed
than
which objects are not needed
anymore
in program analysis!

Use Cases

benchmark	LoC	tick	refresh	free	aux	total
mpg123	16043	1	0	(-)43	0	44
JLayer	8247	1	6	0	2	9
Monte Carlo	1450	1	3	0	2	6
LuIndex	74584	2	15	0	3	20

Table 2. Use cases of short-term memory: lines of code of the benchmark, number of tick-calls, number of refresh-calls, number of free-calls, number of auxiliary lines of code, and total number of modified lines of code.

self-collecting mutators

space overhead

C: Memory

original ptmalloc2

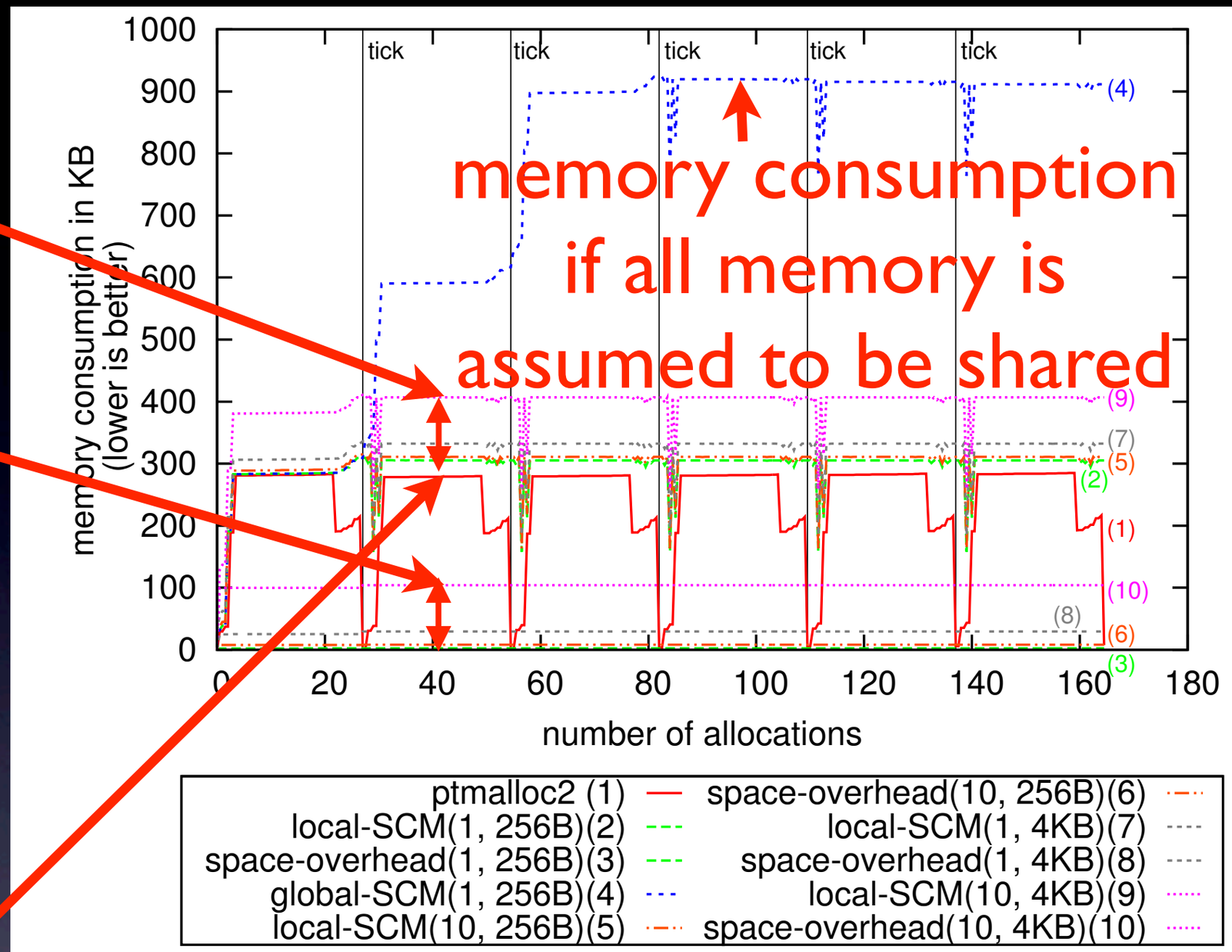


Figure 15. Memory overhead and consumption of the mpg123 benchmark. Again, local/global-SCM(n, m) stands for self-collecting mutators with a maximal expiration extension of n and descriptor page size m, using local/global-refresh. We write space-overhead(n, m) to denote the memory overhead of the local-SCM(n, m) configurations for storing descriptors and descriptor counters.

Java: Throughput

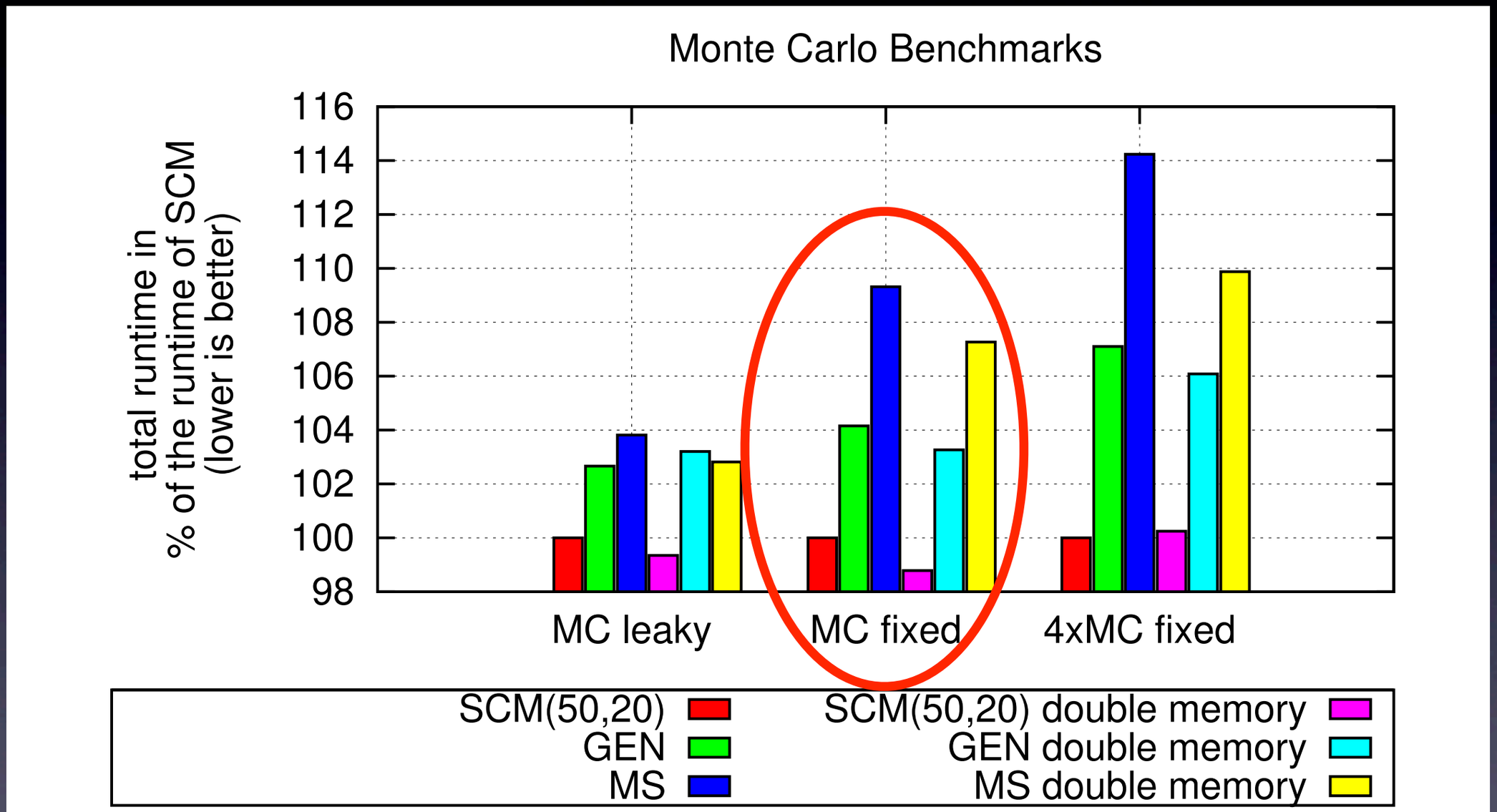


Figure 9. Total execution time of the Monte Carlo benchmarks in percentage of the total execution time of the benchmark using self-collecting mutators.

Programmable Temporal Isolation

Do We Need Programmable?

- Variable-bandwidth servers (VBS) [SIES09]:
  - a process is temporally isolated if the variance in response time of any given piece of process code is bounded independently of other processes
  - response time (throughput) and variance (latency) are programmable at runtime
  - lower variance means more overhead [RTAS10]
- C code available at:
  - tiptoe.cs.uni-salzburg.at/scheduler

Probably yes

But only for uniprocessors...

- Variable-bandwidth servers (VBS) [SIES09]:
  - constant-time scheduling algorithm
  - queue management plugins trade off overall time and space complexity:
    - from linear time (# of processes) and constant space to constant time and quadratic space (timer resolution)
- constant-time admission test:
  - false negatives vs. more overhead
- what about I/O?

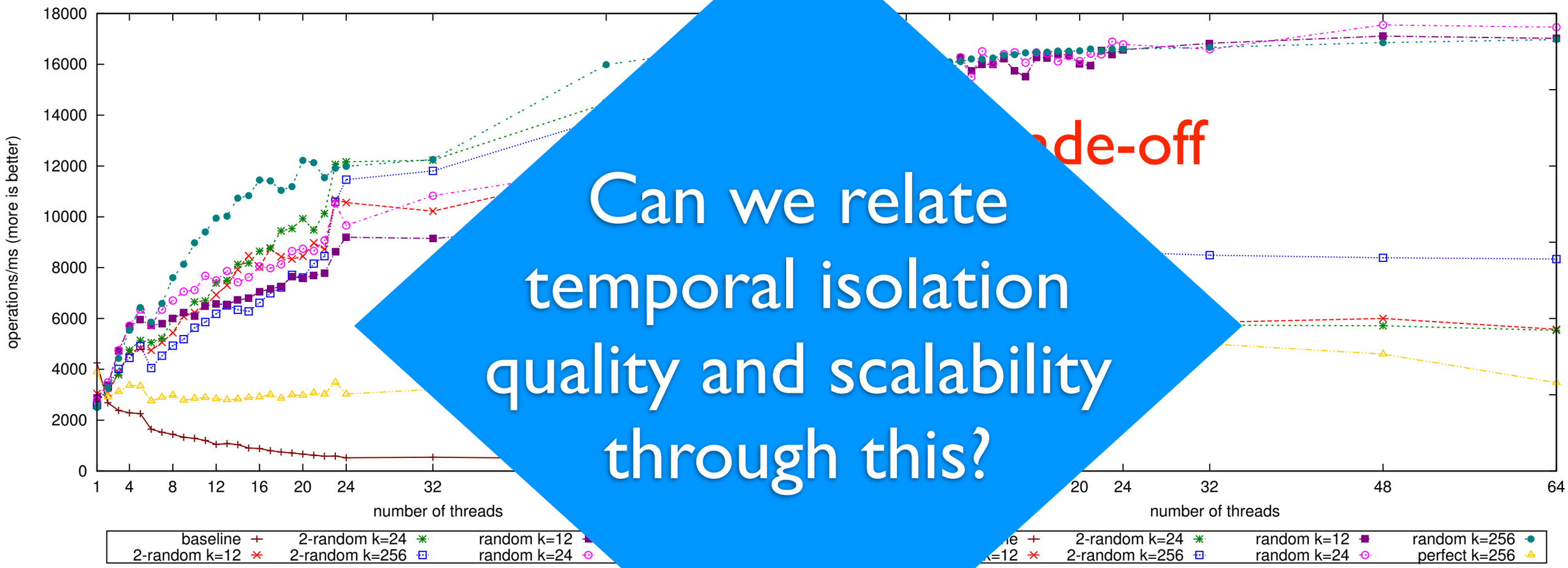
Can We Scale This?

- VBS may likely support multicore, see other work
  - however, what exactly is the relationship of temporal i… …uality, cost, and scalability?
- Non-li… … holds a preliminary answ… …can
  - wh… …s for scalability?
  - shift… …nalysis
  - e.g.:… … queue may only return *k*-… oldest element but scales like hell
- Code not yet available but will be, stay tuned

works with data structures that have an "insert-remove API", e.g. stacks, queues

Lock-free FIFO Queue

(on 24-core machine)



(b) lock-free FIFO

(b) lock-free FIFO

Can we relate temporal isolation quality and scalability through this?

Scalability

Semantics

Time and Power Isolation

Time and Power

- temporal isolation if and only if power isolation?
  - probably yes, if there is no frequency scaling, and if scheduling and context switching cost (time [RTAS10], power?) is accounted for
    - problem: false negatives; solution: PA!?
- power-aware temporal isolation [EMSOFT10]
- time and power isolation w/ frequency scaling?
  - problem: non-linear relationship of power consumption and processor frequency

Thank you

Check out:
eurosys2011.cs.uni-salzburg.at