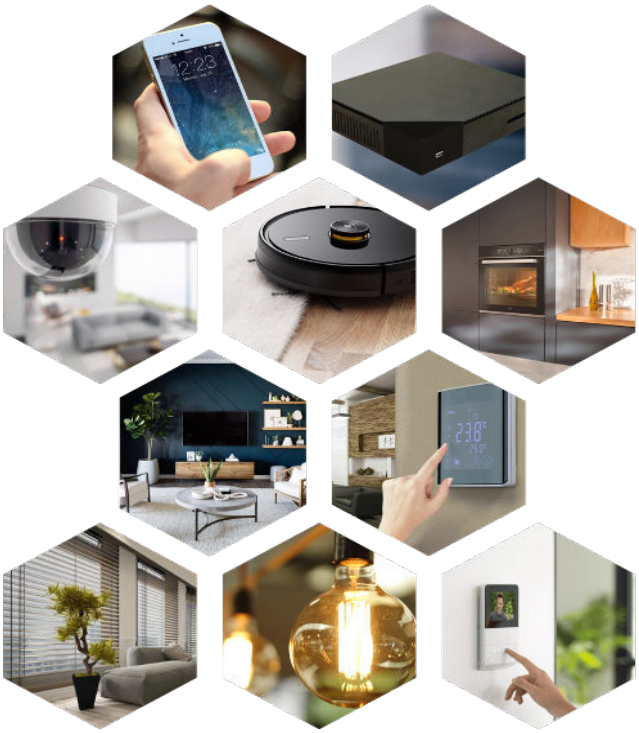
IOT COMPUTING SHALL REQUIRE ANOTHER PROFOUND OS ARCHITECTURE RESET

HyperPanel Lab



Low Energy OS

Secured by design determinist OS

Hybrid OS RTOS & GPOS

Hardware & Telecom agnostic

Scalable OS

Distributed OS

Platform OS

Sovereign OS

None of the existing RTOS (Real Time Operating System), neither any of the GPOS (General Purpose Operating System) kernels dominant architecture could alone tick all of the essential requirements that IoT computing shall entail.

Only a truly hybrid new GPOS, being both RTOS capable (i.e.: fast, determinist and secured by design) - while also remaining lightweight, ultra-low power, reliable over decade long periods - could fulfil the upcoming challenges arising within the Internet of Things context (IoT).

HyperPanel Lab has developed and fully copyrighted such a candidate Hybrid GPOS kernel disruptive architecture for the IoT Computing purpose - HPL-OS4.0

INTRODUCTION

The Personal Computing revolution has durably fostered Microsoft Windows' OS domination. Then, Mobile and Cloud Computing leveraged instead the UNIX OS and all its derivatives (being Linux, Android and iOS). Similarly, the Internet of Things (IoT) upcoming era shall again necessarily require another profound OS architecture reset.

Typically, within the scope of the Internet of Things, software defined vehicles and Industry 4.0 automation will be triggering the quest for a whole new hybrid OS being both in a row an embedded RTOS (Real Time Operating System), as well as a computing GPOS (General Purpose Operating System). Equally challenging, Consumer based Smart IoT devices will also require - quite often within a rather limited hardware footprint - up to decades long seamless support and native security robustness.

In a nutshell, a RTOS has been for decades the preferred OS of choice for developing embedded computing devices and systems whose features rely on real-time data, ultra-fast and predictable context-switching, and determinism to deliver an accurate output within an expected timeline.

But RTOS do lack, among other things, advanced user interface capabilities, as well as APP based versatility, adaptative connectivity, and multitasking modern functions and throughput that only a GPOS can enable. However, incumbent GPOS could, by no means, support the strictest time, preemptive mode and determinism boundaries that only RTOS can natively perform for the purpose.

IoT OS SPECIFIC ISSUE, AS IT STANDS

Beside the inability to run on lightweight memory footprints and power limited MMUs, it is essentially the lack of real-time and secured performances that remains the unsolved critical impediment for a GPOS wider adoption within the IoT ecosystem, then the RTOS phase out.

Within the IoT context, and beyond real-time responsiveness, RTOS will therefore remain here to stay because of their still unrivalled ability to meet most Things' Computing essential requirements such as deterministic task handling and optimal resource usage, ultra-low power consumption and even year's long battery-based usability, not to mention data extracting reliability and trust.

Today, software defined vehicles are the typical IoT connected platforms where RTOS have to cohabit with GPOS in a sometimes rather complex combination. Reason is that RTOS are the only solution to propel high-priority security functions, such as for example the automatic air-bag control system. Indeed, as soon as the vehicle senses a shock, the airbags should be activated within a milli-second whatever a smart GPOS may be executing in the meantime for the cockpit control interface, the self-driving mode, or the infotainment. Hence, being able to preempt the kernel and enforce critical task scheduling to dispatch threads and processes onto the CPU is typically what GPOS lack.

On the other hand, GPOS can indeed perform a much higher overall throughput than any RTOS could ever come anywhere close to, which is precisely what desktops, smartphones and server applications do require. As such, GPOS enable dynamic memory mapping and random execution patterns, but in no circumstances, they could match RTOS's native ability to be deterministic (but having no random execution pattern), within a predictable response time, time bound and preemptive kernel.

INTERMEDIATE SOLUTION - TRADE OFF

As noticeably stated by Paul Leroux, a former OS expert at Blackberry / QNX, "*this is not a matter of RTOS good, GPOS bad. GPOSs such as Linux, Windows XP, and UNIX all serve their intended purposes extremely well. They only fall short when they are forced into deterministic environments they were not designed for, such as those found in automotive telematics systems, medical instruments, and continuous media applications*".

In essence, since GPOS like Linux do not have a preemptible kernel, then even a high-priority user thread could never preempt a kernel call, but instead,

it has to wait for the entire call to be completed – even if that call was invoked by any of the lowest-priority process. Such behavior obviously causes unpredictable delays and prevents critical activities from being executed on time (which again, is definitely not acceptable for an airbag control system).

There are currently 3 approaches at stake to try and solve that dilemma between RTOS and GPOS for the purpose of IoTs. Here they are:

A => One is to try and expand the RTOS original capabilities in terms of user interface to try and come closer to the demand for increasingly sophisticated Graphical User Interfaces, 3D rendering, and web based advanced graphics that are popular in most GPOS. Indeed, it is then important to keep on isolating as much as possible the user interface from the rest of the RTOS process to keep fulfilling the strictest certification requirements, such as EAL5 and other Industry specific security Standards for avionics, automotive, medical, etc.

B => Alternatively, there has been many attempts to try and improve task preemption of existing GPOS through the development of a number of real-time extensions, tasklet and patches.

But in practice, this comes with a price in terms of porting complexity and maintenance efforts. In practice, it usually comes down to a dual-kernel approach whereby the GPOS runs as a task on top of a dedicated RTOS. Then, critical tasks that require deterministic scheduling run in the sub kernel and can therefore preempt the GPOS. However, such tasks cannot rely on any of the existing GPOS mainstream services, unless being subject again to the same preemption problems that prohibit GPOS processes from behaving deterministically. So, additional new drivers and services must be created specifically for the real-time kernel itself – no matter if those already exist within the GPOS - thus creating additional fragmentation and maintenance issues. Also, coding complexity is increasing in several order of magnitude as typically most existing GPOS drivers are not preemptible. So, to foster predictability, one must insert numerous preemption points into each drivers.

C => The last option, recently popularized by Google (with Fuchsia OS and Kata OS), but also Meta (with the now discontinued XR/OS project), and even Huawei (with its HarmonyOS, yet to be released, roadmap) is to adopt a micro-kernel type architecture.

As per a common description, "*in a microkernel, only a small core of fundamental RTOS services (such as signals, timers, and scheduling) reside in the kernel itself. All other components (such as drivers, file systems, protocol stacks, and applications) run outside the kernel as separate, memory-protected processes*".

Doing so, the microkernel contains only the near-minimum amount of code that provides the base mechanisms required to implement an OS on top. Therefore, it overcomes the issue of RTOS piloting a GPOS, whereby if a real-time task contains a coding error then it can cause a fatal kernel fault. Instead, a microkernel does contain memory protection features and it can be broken down into separate processes called servers. In other words, the kernel and the user services are isolated, so if any of the user services fails, then the kernel service remains unaffected.

Microkernel have been getting renewed traction because unlike patches and attempts to make a GPOS real time, it significantly streamlines the development of the specific drivers and extensions necessary to support a GPOS as a service. Also, Microkernel are scalable as new services can be added to the user address space, without changing the kernel. However, as the drivers have to be implemented as procedures, a context switch, or a function call are required.

IoT COMPUTING IS YET TO BECOME COLLABORATIVE, THUS ENTAILING A WHOLE NEW OS ARCHITECTURE RESET

Internet has evolved from originally connecting people to people, and then people to physical things, to eventually connect physical things to other physical things, all seamlessly and in real time. This is called the Internet of Things (IoT).

Then, without any human intervention, IoT smart devices collect, send and act on the data they acquire from their environments, including the information they get from one another. So, after Personal Computing - and then next Mobile Computing - time has now come for Things' Computing to happen.

But the same way both Computing and Internet have gone through drastic mutations over the years, so will the Internet of Things as well.

Consequently, we are just at the beginning of another truly disruptive Computing & Internet major Revolution, and as always, this will require another profound hardware (microprocessor) and software (Operating System) reset.

Originally, Internet was purely about basic connectivity, thus digitizing our connections (emails, web browsing, … alongside the emergence of global champions such as Google and Yahoo!). Later, in a second phase, Internet digitized both the business and the process (i.e.: it is the eCommerce boom with new champions being Amazon and Alibaba).

Then, Internet unleashed whole new collaborative and immersive experiences, thus digitizing all of our interactions, both private and business. That has been the time of social media, video streaming, eSport, Metaverse, Cloud containerization & AI as a Service, ... with new iconic stakeholders being Facebook, Netflix, etc.

The Internet of Things will also come along the same 3-steps evolution path – i.e.: connectivity first, then business transformations, and ultimately, collaborative computing.

At first, and till now, IoT is still very much about nascent connectivity, which is the reason why the US and China consider the Cloud and 5G so crucial. Next, IoTs are set to truly revolutionize smart cities, transportations, industries, agriculture, heath, … the same way eCommerce platforms like Amazon have revolutionize our traditional retail. Then, eventually, in an ultimate and third stage, IoT will be entering into the genuine collaborative Things' Computing era (like Social Networks have profoundly affect people interactions).

That ultimate IoT collaborative phase shall obviously raise fundamental and whole new technical challenges that none of the current Computers, Smartphones or Routers incumbent hardware, neither any of their mainstream Operating Systems (OS), will be capable of solutioning alone.

The new desired OS of choice to truly unleash the collaborative IoT computing future revolution will necessarily have to be hybrid – i.e.: both RTOS and GPOS capable – as well as lightweight, ultra-low power, secure and reliable over decade long periods.

While there are plenty of RTOS and microkernel available, the GPOS only come down to 2 architectures, both under US copyrights. One is indeed Microsoft, and the other one is UNIX whose routes and derivatives are Linux, Android, MacOS, iOS, Solaris, IBM AIX, …

That is where HyperPanel Lab comes in the play with a compelling, and absolutely unique, level playing field turnkey and readily available OS kernel solution.

HyperPanel Lab is a French computer science company founded in 1988, whose founders and world-class OS experts have created from scratch, over 25 years of self-financed R&D, a 100% sovereign and fully copyrighted GPOS breakthrough architecture.

Beyond being the sole and unique third GPOS architecture available to date - beside Microsoft and Unix - HyperPanel's OS is the sole and only GPOS that, within a very small footprint, unleashes the full benefits of both a deterministic RTOS as well as the genuine multi-tasking capabilities of a modern GPOS, but in a modular and distributed architecture.

HPL-OS4.0 comprises 2 layers being a software extension of the von Neumann hardware architecture to the Operating System itself. The upper layer deals with the applications, while the other one manages the hardware – i.e.: handling both the I/O and the telecom stacks. This lower layer is therefore acting as a kernel having hands' on all the drivers, communication protocols and memory access.

The technology is leveraging the Turing Machine principles as FSM (Finite State Machine) are implemented for all the drivers and telecom protocols which are then operated by the dedicated FSM engine. Using FSM enables an ultra-efficient interruption-based monitoring of the kernel, thus allowing to drastically lower latency, while matching as close as possible the hardware fastest theoretical processing time.

HPL-OS4.0 kernel has been entirely developed in France over 2 decades, in a clean-room R&D environment, by HyperPanel Lab's OS architects. The company controls 100% of the copyrights and therefore does not fall under any GPL license scheme, nor any disclosure obligation of any kind, and neither do its licensees. It also means that the Kernel lifecycle is under full control and does not depend upon any external provider, nor any foundation.

Offering stunning real-time execution speed and latency, instant boot, ultra-low energy consumption, DRAM-less use cases, and seamless updates while running ; HPL-OS4.0 kernel comes with another unique characteristic: its native ability to dynamically distribute, without any latency, all of the IoT's hardware resources amid propelled devices connected within the same edge Cloud (which is another benefit of the smart application of the von Neumann machine theory).

With millions of devices already deployed by tier one operators, OEMs and CE manufacturers, the Technology concept has already proved its unrivalled ability to deal with complex video and multi-telecom protocols and secured by design ecosystems. It is immediately available with a compelling source code comprising both the specification and the formal verification unitary tests. Additional academic type Computer Science papers are also available under NDA to further ease the technology hand-over process.

HYPERPANEL'S HYBRID HPL-OS4.0 OS TECHNOLOGY ABSTRACTS

HPL-OS4.0 Key Features

- Collaborative IoT computing & M2M will involve new edge type 4.0 routers capable of handling seamlessly hundreds of devices & sensors in a row with massive data throughput and real time interrupts that HPL-OS4.0 four-level of execution architecture is specifically tailored for.
- HPL-OS4.0 is a hybrid OS, being simultaneously GPOS (General Purpose Operating System) and RTOS (Real Time Operating System).
- HPL-OS4.0 is agnostic with respect to telecommunications modes and can handle simultaneously dozens of different telecommunication protocols - wired or wireless, broadcast or with return path - in a seamless manner.
- HPL-OS4.0 is a true distributed OS that allows sharing all the hardware resources of any objects equipped with HPL-OS4.0 and connected on the same edge network. All these objects can then be seen as a single virtual object, with a multicore whose internal bus is the local network itself.

HPL-OS4.0 Fundamental Innovations

- A complete Operating System (OS), including:
  - Its own Kernel (HPL-OS4.0 Kernel).
  - Its own Middleware (HPL-OS4.0 Middleware).
  - Boot & Loader (HPL-OS4.0 Boot & Loader).
- A Kernel architecture derived from John von Neumann's computer architecture with a clear separation between:
  - Inputs/Outputs, Protocols & Drivers (I/O container).
  - And applications processing (Application container).
  - The 2 containers communicate only by messages.
- A Kernel powered by a technology derived from the Turing Machine:
  - All drivers and protocols are developed in the form of Finite State Machine (FSM), themselves powered by a dedicated FSM engine.
  - The same approach can be used for the application through a dedicated FSM engine.
- A Kernel's Scheduling of tasks based on hardware interruption.
- An OS ready to be industrialized as the source code - in C language - includes:
  - Embedded documentation, and comments.
  - Built-in self-running unitary tests.

HPL-OS4.0 DISRUPTIVE ARCHITECTURE OVERVIEW

HPL-OS4.0 is the first Finite State Machine based GPOS, thus bringing native determinism.

The required memory footprint can have a very wide range and can be as low as 150 KB of Flash and 80 KB of RAM, which is absolutely unique for a GPOS.

The code remains stored in the FLASH memory and does not need, neither does the context, to be copied in RAM in order to achieve ultimate execution performances. Hence, the RAM memory is only used to store variables. When the code is located in FLASH and is not executed / used, absolutely no energy is consumed - whilst any code storage in RAM draws by itself a need or a larger RAM, which drastically impacts energy consumption and IoT cost.

HPL-OS4.0 comes with 4 levels of execution. This allows HPL-OS4.0 to be fastest than most RTOS to date, whilst still offering complete GPOS functions and capabilities. These four levels of execution are spread into two specifics containers: one being in charge of managing the I/O, and another one being in charge of managing the applications.

Tasks are handled as FSM, using a system dedicated transition table. The task scheduler is a subroutine of a more general engine. Task semaphores are implemented using system events.

Every transition is associated to two parameters, which are the event which causes that transition, and the treatment which is run.

HPL-OS4.0 suppresses all hardware to software latencies means that whatever hardware event arises (i.e.: an interrupt), its software processing is immediately started within one CPU cycle (interrupt handler). Then, the next processing steps (pipe) will also be started at the next CPU cycle. To achieve that, activity switching time has to be reduced by a ratio between 100 to 1.000. Experiments and computations are showing between 400 % to 1.000 % benefits in terms of I/O throughputs increase with the very same hardware.

That sub-level container is based on Finite State Machines (FSM) to better echo any processors' ultimate hardware capabilities, while also enabling secured management of the Inputs & Outputs - both in terms of process, memory access and data handling – as well as the Protocols. Whereas GPOS are all monolithic, HPL-OS4.0 isn't: it is at last truly modular and scalable, somehow likewise microkernel based architectures aim to be.

The HPL-OS4.0 two containers are both leveraging their own sub-systems and operating independently one from another, thus enabling several execution levels and time endorsement capabilities.

1 => At low level, a first container handles all Inputs/Outputs (I/O), the system resources and the telecom protocols. Thanks to the FSM, the treatments become deterministic, bringing security and speed. It also enables to manage these fundamental tasks at unrivalled speed, up to 40 times quicker than any GPOS could, and up to 4 times faster than most RTOS would.

2 => At upper level, the second container manages the applications, like any GPOS would, typically the JavaScript interpreter and the HTML browser. This container has its own dedicated real time control unit, and it can even be interfaced with a third-party OS such as Linux, thus allowing easy re-use of legacy Apps and software developments.

These 2 containers communicate in an asynchronous manner using an optimized messaging mechanism. They can therefore run side by side on any multicore processors, or separated on different physical IoT devices running as if they were one single object dynamically sharing hardware and functional resources at the edge, rather than just data and files as usual.

This makes HPL-OS4.0 a truly shared and scalable hybrid GPOS perfectly tailored for the new IoT's empowered paradigm. The software package – coupling both the OS with its secured boot and tailored-made loader - also enables to achieve the best ever-possible trade-off in terms of hardware loads (both memory and CPU), native robustness and security, autonomy, performances per MIPS, and above all, fundamental "security by design" IoT expectations.

BACK TO BASICS

The OS is the essential foundation software that manages both the hardware as well as the other software, thus providing system resources to meet the demands of all the programs and applications.

Technically speaking, GPOS are designed to perform multiple tasks running at the same time, while RTOS are tailored for time-sensitive applications requiring determinism.

But generally speaking, whereas a GPOS is perceived as the actual interface between the user and its device, it is actually down to the OS Kernel subset to provide underneath the necessary interface between the software applications and the hardware. As such, the Linux Kernel is the foundation for the Android GPOS.

Hence, one could assert that the OS Kernel is somehow to a General Purpose OS, what the ARM or RISC-V processor IP blocs (i.e.: the CPU core) are to any General-Purpose microprocessor.

In other words, while there are plenty of GPOS products out there (Android, iOS, Windows, Ubuntu, Tizen, …), as well as plenty of General-Purpose microprocessors (Pentium, Intel Core, Xeon, Qualcomm Snapdragon, STM32, Broadcom BCM, …), there are actually a very limited number of OS Kernels and CPU architectures sustaining them.

CISC is typically the CPU architecture of choice for both Intel and AMD, and RISC, the base Kernel architecture implemented by ARM as well as RISC-V.

Similarly, side to Windows which leverages the Microsoft original kernel, it is the UNIX Kernel architecture that supports all of the other incumbent GPOS, including Linux which is a custom implementation of UNIX, the same way ARM is a proprietary implementation of RISC.

So, considering that commonly speaking, a GPOS is often perceived as a branded and application specific OS based on a given kernel architecture (such as ANDROID), HPL-OS4.0 is not to be ranked as a commercial GPOS. Instead, HPL-OS4.0 is echoing UNIX and Microsoft architectures, thus providing the third core kernel of choice that makes it possible to easily add on business specific services to develop a turnkey GPOS (the same way ANDROID adds data storage, screen display, multimedia, and web browsing to the Linux Kernel).

However, HPL-OS4.0 is to date the only one of the three OS Kernel architectures at stake capable of seamlessly running a RTOS-capable hybrid GPOS for the Internet of Things, thus combining both advanced multitasking GPOS capabilities, together with lightweight, ultra-low power, security by design and determinism RTOS features.