TECH PAPER

# Linux System/User Space Virtualization
For Next Generation Real-time Embedded Applications

**Enea has initiated and is co-maintaining a "Linux meta-virtualization layer" within the Yocto Linux build environment consisting of both system and user space virtualization technologies for multicore devices. Specifically the program consists of: a) to collaboratively research/benchmark LxC/KVM/Xen virtualization combined with advanced core isolation techniques, and b) to integrate and contribute OpenFlow, Open vSwitch, CRIU, and DMTCP along with incremental contributions of OpenStack components. Traditionally most of these Linux initiatives have primarily been associated with enterprise level Linux applications for Cloud Computing, not embedded computing. So how can these possibly relate to the embedded systems world that Enea and its solutions are about? This article will show how the "so-called" embedded landscape is rapidly changing, i.e. the lines between what is considered embedded computing and enterprise computing are blurring. Given this, the role of these virtualization technologies in the newly emerging embedded applications space is summarized. This article is not a primer on the selected technologies – the reader can look them up – but it shows the motivation for usage of each in this new embedded world.**

Historically, the concept of embedded computing has been around for 40 some odd years. The world's first small real-time executive was created for the NASA Apollo program, but minicomputers ruled the day then. Embedded computing really started to take off with the advent of small micro-controllers/processors in the late 1970's and early 1980's, hence the rise of many Real-time Operating System (RTOS) kernels in that period. Originally, embedded computers were deployed in relatively simple control applications, but after a while it became apparent that distribution of embedded devices, i.e. offload of the tasks of one processor into many would be needed for the expanding roles of such systems. This then drove the requirement for increased connectivity, namely the focus on interconnect and communications protocols and as witnessed by the rise of IP protocols in the 1980's, and continuing into the 1990's. But then another trend started, namely the computing concept of "control/management" versus "data or user" plane processing. Control/management processing applications often are more about throughput performance versus real-time responsiveness (the concepts are not equivalent, see the Enea white paper on Real-time Linux). Originally, many embedded applications had to deal with both performance/throughput and real-time responsiveness, and the result was even more distribution and specialization of functionality for a given system. But in all of this, one factor remained consistent, i.e. that all truly embedded applications/systems had a closed, fixed network or distributed system architecture composed of discreet hardware units for all the layers of needed functionality (with the added advantage of isolation from external impacts). One can see this especially in telecom/networking systems as they have evolved over the last 20 years, as well as in the medical, industrial, and non-telecom communications industries or applications. If one wanted to increase capacity of a given system, then the simple answer was to install more connected nodes to the network.

But in recent years, something new has happened. The exponential growth of the capacity (bandwidth/throughput and real-time responsiveness) and availability of access to the internet, has lessened the requirement for complete isolation of a fixed hardware distributed, embedded real-time system. That is, some aspects of control processing do not have to be co-located with the real-time or data/user plane processing components because they can be accessed remotely. Even so, this is not the driver for separation of certain aspects of control/management from the data/user plane aspects of a complex system. It is but an enabler. The real driver is that in the last 10 years, we have seen the rise of a) complete connectivity between many embedded devices and higher level control/management systems, i.e. "internet of everything", and b) the exponential increase in the requirements of exchanged data that is required so that embedded devices can become “more intelligent”. The world is becoming transformed by the concept of “big data”, wherein more and more data is required to manage the complexity of interconnected things that enables the concept of “more intelligent devices”. The heretofore traditional model for embedded systems to meet this demand would simply be to add more physical computing nodes to the system to handle the communications bandwidth requirements. But in the modern world, this model does not scale because for service providers whatever the industry, CAPEX and OPEX costs that have to be passed to the subscribers or users of the service become prohibitive factors for the massive and continuing expansion of the bandwidth that the services will require. This means that a new model needs to be adopted for traditional embedded applications. Ok, perhaps this sounds maybe a little too abstract. So let’s take a couple of use cases from the telecom and medical worlds to illustrate the point.

## NEW PARADIGMS FOR EMBEDDED – MOBILE INFRASTRUCTURE AND MEDICAL DEVICES

### Mobile Infrastructure – Radio Access Networks (RAN)

Traditional Radio Access Networks (RAN) are one of the very best examples of embedded real-time systems. Simply RAN networks primarily consist of stand-alone base stations (BTS). Each BTS covers a relatively small area. The well documented exploding and exponential demand for more bandwidth means that there is tremendous focus by operators and equipment manufacturers to improve both coverage (more subscribers per unit area) and processing bandwidth capabilities of the BTS system. There are many potential solutions involving adding more fielded equipment and increasing bandwith of each BTS like macro-cell, metro or micro cell, pico-cell, femto-cell, improved remote radio access towers separated from the BTS’s, and other orthogonal solutions such as Wi-Fi offload. All of these potential solutions have some validity, but this article does not intend to address the merits, demerits, and trade-offs associated with such. The important point in all of this is that traditional cellular systems have significant limitations. First, each BTS is costly to build and operate. The supporting facilities cost of any BTS are a significant factor that doesn’t improve much with additional deployment of physical nodes regardless of size, coverage, even with greater processing bandwidth for each BTS. Secondly, because mobile users are moving from one place to another and also because of the natural burst of mobile data applications, the traffic of each BTS fluctuates very much from time to time, day to day. The average utilization rate of individual BTS is very low. However these processing resources cannot be shared with other BTS. Thus all the BTS must be designed to handle the maximum traffic expected no matter the size of the average traffic. This causes a lot of wasted processing resources and therefore cost to the operators trying to handle the demand. So the real issue from an operators perspective, is not just how to solve the bandwidth and reach for subscribers for today, but how to build a solution that optimizes processing capability such that it will scale to the ever increasing future requirements for such with reduced or at least manageable costs (CAPEX and OPEX). Clearly, offloading or replacing fixed processing capability in the field with the more dynamic and elastic processing capability of a collection of localized Cloud servers can help achieve these operator goals. But Cloud computing is generally associated with enterprise computing and not embedded computing. So the lines between enterprise and embedded computing are becoming blurred even in the traditional telecom space.

### Medical Devices – Patient Monitoring and Treatment

Next generation patient care is becoming increasingly intelligent and connected. For example, take drug infusion devices. The evolving architecture of such systems has centralized servers with access to patient data, and that are connected to a collection of intelligent infusion pumps. The number of pumps may be in the many thousands, so a network of intermediate “aggregator” devices is deployed to offload the servers to reliably maintain physical connections and control and status communications to a smaller collection of pumps in a local area. These aggregator devices are traditional “embedded” devices in every sense. However, the physical deployment of the pumps is very dynamic, i.e. they can be moved frequently and be on/off at any given time. This means that there could be a natural imbalance of workload across all the aggregators and this is in turn reflected in inefficient utilization of the collection of aggregators. If some or much of the aggregator functionality can be moved to a centralized Cloud architecture that can elastically adapt to the processing demand, then tremendous cost savings can be achieved by reducing the total number of fixed aggregators that are required. Now add even more device types like cardio/blood monitoring and other patient monitoring equipment in a centralized integrated medical care infrastructure that means support of tens of thousands of connected devices. The overall network processing requirements of such a large number of managed devices grow even more complex, unwieldy, and costly. Adding Cloud computing offload in the integrated medical care environment is yet another example of the blurring of the lines between embedded computing and enterprise computing.

## WHY THEN SYSTEM/USER SPACE VIRTUALIZATION?

So what does this shift in the embedded computing paradigm have to do with the subject of this article, namely Linux System/User Space Virtualization? Cloud computing offload of some embedded system functionality really only makes sense when one wishes to scale to large networks of managed embedded devices. This has two consequences: 1) virtual environments in the Cloud are best for optimization since the whole environment can be dynamically configured for best characteristics under load, and 2) it is best to break the entire managed network into smaller separate domains that offers better manageability, maintenance, and isolation/security across domains (e.g. for different user classes, different geographical locations, etc) and also better performance since load from one domain should not affect other domains. The best way to create sub-domains in a Cloud server virtualized computing environment is to create virtual networks that connect each environment to the embedded managed device domains. This kind of architecture helps achieve the designer goals for next generation massively connected device networks, namely scalability in terms of easy system expansion, configurability or re-configurability, and performance optimization, and this dramatically lower operator CAPEX and OPEX when much of the processing capability is co-located in secure physical environments such as a collection of Cloud servers.

So let’s get back to the Yocto Meta-Virtualization project. Why the particular selection of components? Technologies such as XEN, KVM, and LxC (Linux Containers) are system level virtualization solutions that offer ways to create and isolate different environments within a single server, i.e. the virtualized computing environments mentioned above. For user space networking virtualization we have OpenFlow for creation of virtual networks within a single physical network and offers sophisticated and traffic management capabilities, and Open vSwitch to connect multiple virtual environments within a server or to other servers necessary for a scalable virtual network environment. OpenStack is not a virtualization technology itself, rather it is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control. So it is an important component of any Cloud based environment. CRIU (checkpoint/restore in user space) and DMTCP (Distributed Multi-Threading Checkpoint), again while not virtualization technologies themselves, are useful user space adjuncts to the above virtualization environments. Applications running in these environments will almost certainly need a high level of reliability with easy restart and restoration of state upon software crashes without the need for much more complex redundancy models in Cloud server based systems.

## THE YOCTO META-VIRTUALZATION PROJECT

So there we have it. Enea with its Linux offering was one of the first companies to achieve the formal Linux Foundation designation of “Yocto Project Compliant” and is helping to further lead the way now with its initiation of the Yocto Meta-Virtualization Project. This is an important step towards providing an open build and configuration environment for any and all who wish to employ the Cloud in next generation real-time embedded system and operations management networks. The project was just started a few months ago and needs more contributors. If you think this is important work for you or your company, then join the project!! You can find out more about the project at the following:

http://www.ohloh.net/p/meta-virtualization

http://git.yoctoproject.org/cgit/cgit.cgi/meta-virtualization/tree/README