tenAsys®

How TSN Can Enable vPLCs and the Software Defined Factory

A Virtualized Approach

Christopher Main & Joel Morrissette

TenAsys Corporation

Florian Frick

ISW, Universität Stuttgart

PLC Evolution

PLCs have evolved in three epochs

Pure hardware PLCs

Simple, cheap, reliable

Limited capabilities and functionality

Dedicated I/O functions

Software PLCs on IPC

More capabilities, extensible

Combine PLC functionality with other software

Dedicated I/O functions

Co-Hosted vPLCs

Consolidated workloads

Multiple PLC functions on one platform

But how do we connect to I/O?

Benefits of Virtual Control

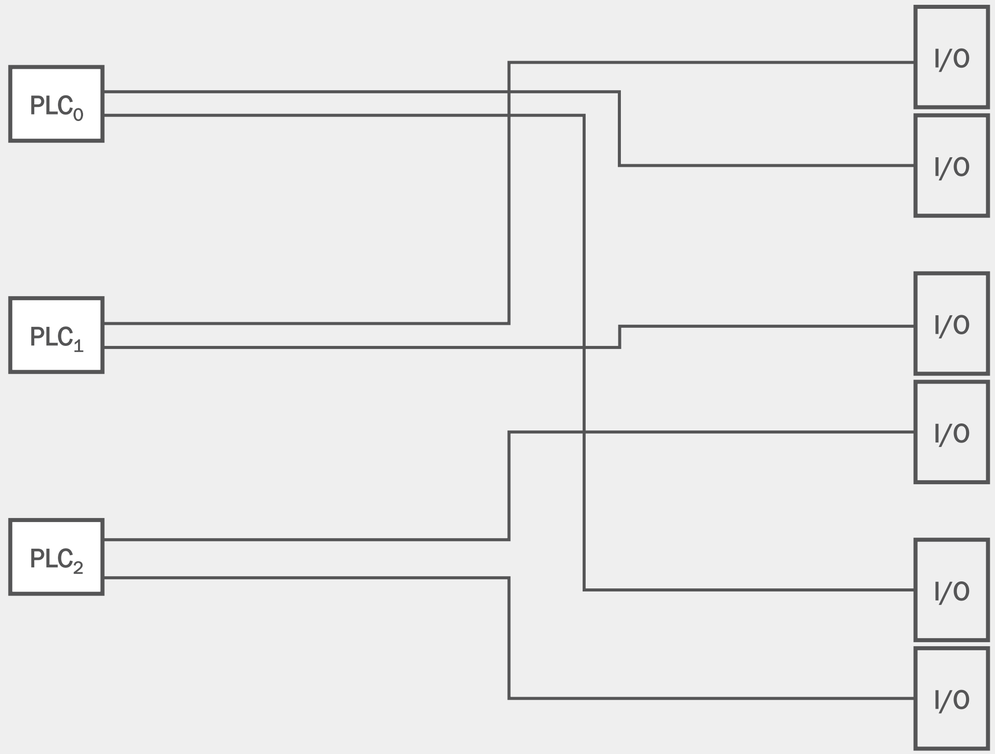
- Easy scaling
  - adding another controller in software is easy
- Lower cost
  - No hardware cost for adding another controller
- Dynamic provisioning
  - Connecting a controller to an I/O may be software-defined
- Easy reconfiguration
  - Easy from the point of view of reconnecting controllers and I/Os, less physical reconnection required

- All important factors in the digitalization of the factory, and eventual SDM.

Integrating vPLCs

Interconnect complexity

- Because there's a direct connection between a PLC and its I/Os, network complexity increases as controllers an devices are added.
- This is a *dynamic* interconnect, which introduces complexity



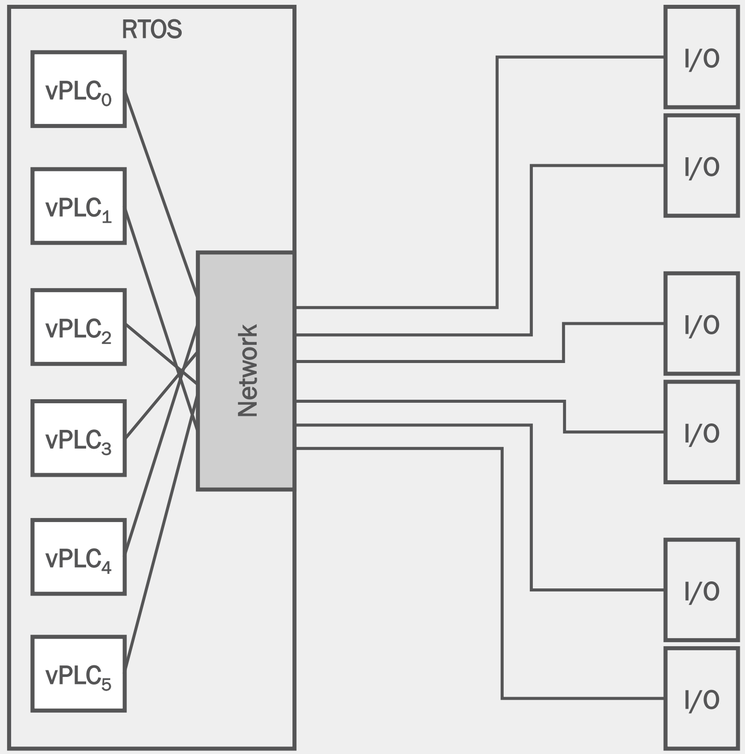
The virtual PLC

- Derived from a Software PLC
  - Runs on an RTOS on an IPC
- Connections to multiple I/Os
  - Typically using one or more Ethernet ports and an industrial protocol stack

- As a vPLC, may be instantiated multiple times as a workload on a suitable platform

Enabling vPLCs using Workload Consolidation

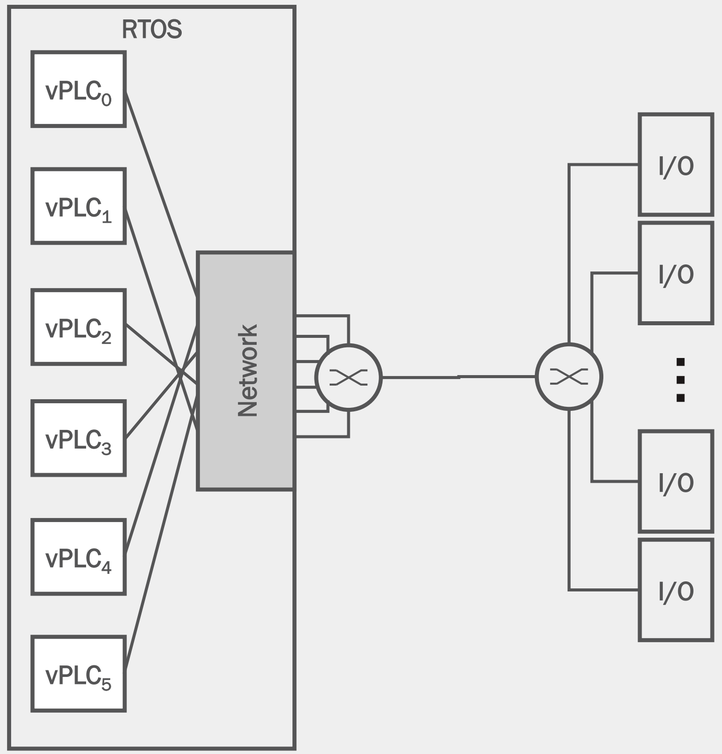
- Initially consider each vPLC as a separate process on a server
- Solves scaling *at the controller* where it's cheaper in resource usage
- Still have an interconnect issue. Increasingly complex and very hard to scale with high node counts.



Solving Interconnect Complexity with Converged Networking

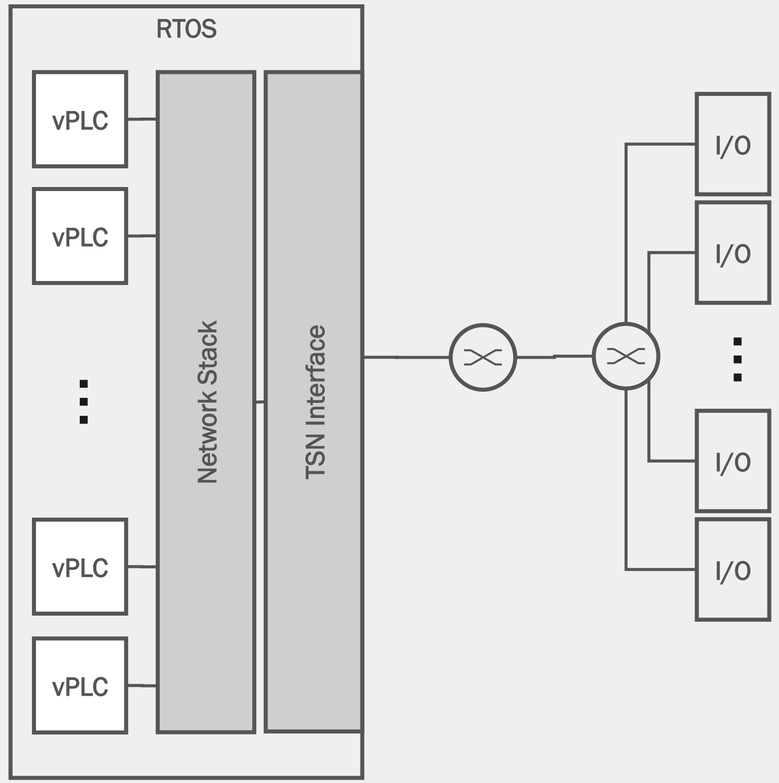
- In the previous example, *every* vPLC needs a network interface
- With Converged Networking it doesn't – the network absorbs the dynamic complexity through TSN

There is still a port routing issue



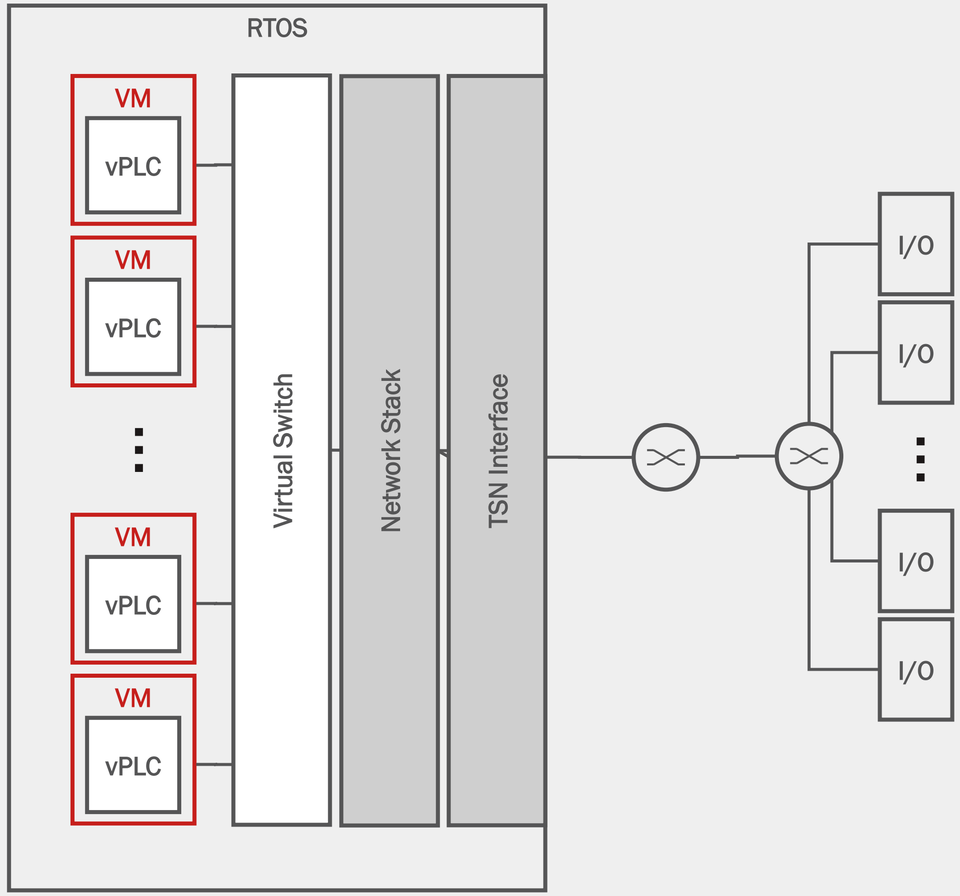
Scaling the Model: TSN interface

- The TSN interface schedules traffic on the host before delivery to the TSN network
- The host becomes a TSN end station subject to synchronization and configuration



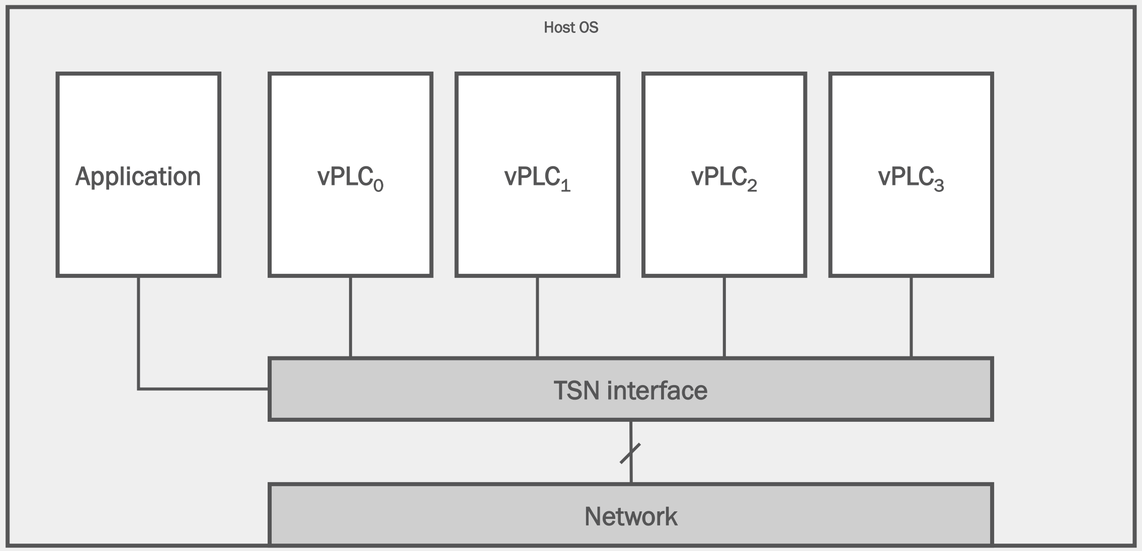
Scaling the Model: Virtual Switch

- A *virtual switch* addresses the complexity on the host before the converged network
- Why a *virtual* switch? It can scale to the high client count and related traffic complexity in a way that a hardware switch may not (and at lower cost).



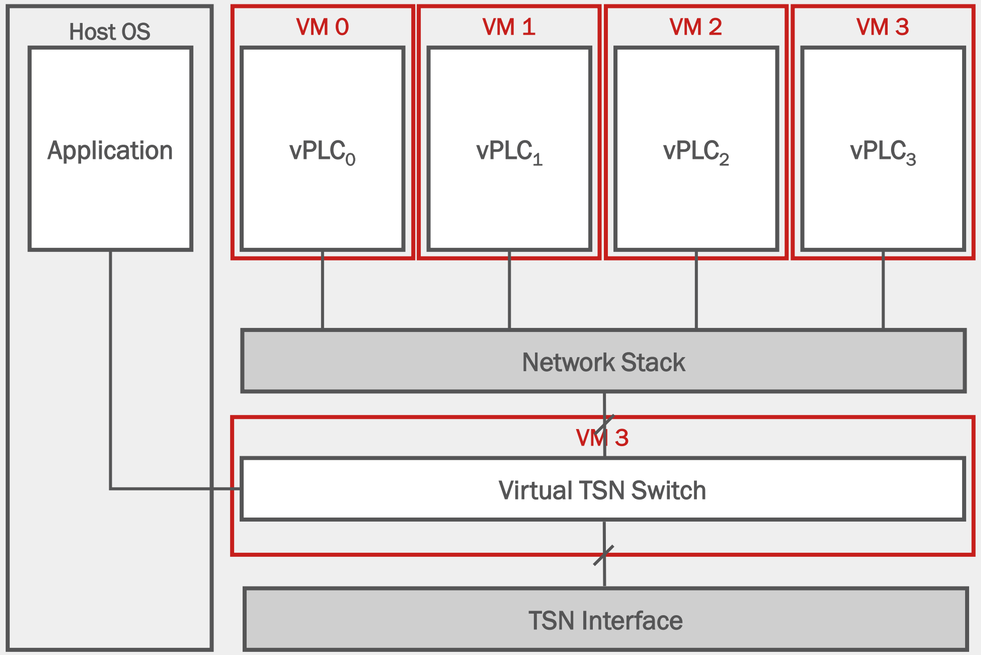
Practical Application Example

- System hosts one or more applications
- vPLCs are connected to the network interface(s) via a virtual switch
- Each application and vPLC is competing for system resources, and latency & determinism requirements must be met



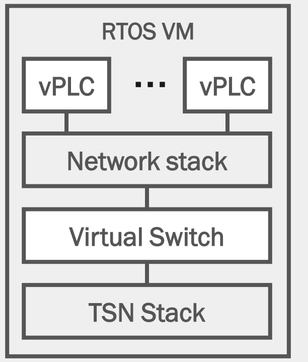
Practical Application Example

- Each vPLC and the Virtual Switch are mapped to VMs that are co-resident with the host OS on the system
- Each VM is now a TSN End Station, therefore configured according to TSN config
- Application continues to run on the Host OS along side VM nodes
- System can be scaled with minimum overhead increase



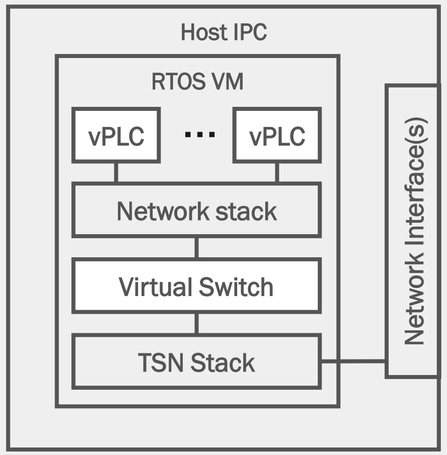
Practical Implementation Example: Prototype

- Multiple vPLCs are hosted within an RTOS VM (INtime in this implementation)
- All network traffic is routed through a virtual switch
- TSN stack schedules, transmits, and receives network traffic



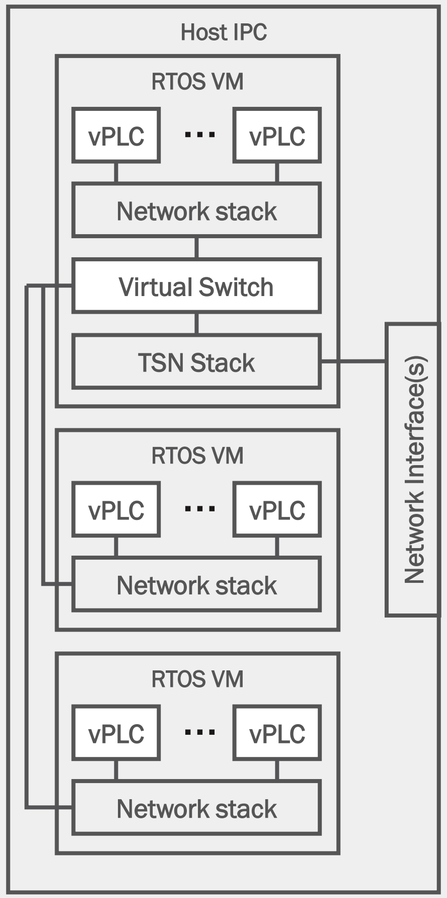
Practical Implementation Example: Prototype

- IPC hosts RTOS VM and provides physical resources incl. physical network interface



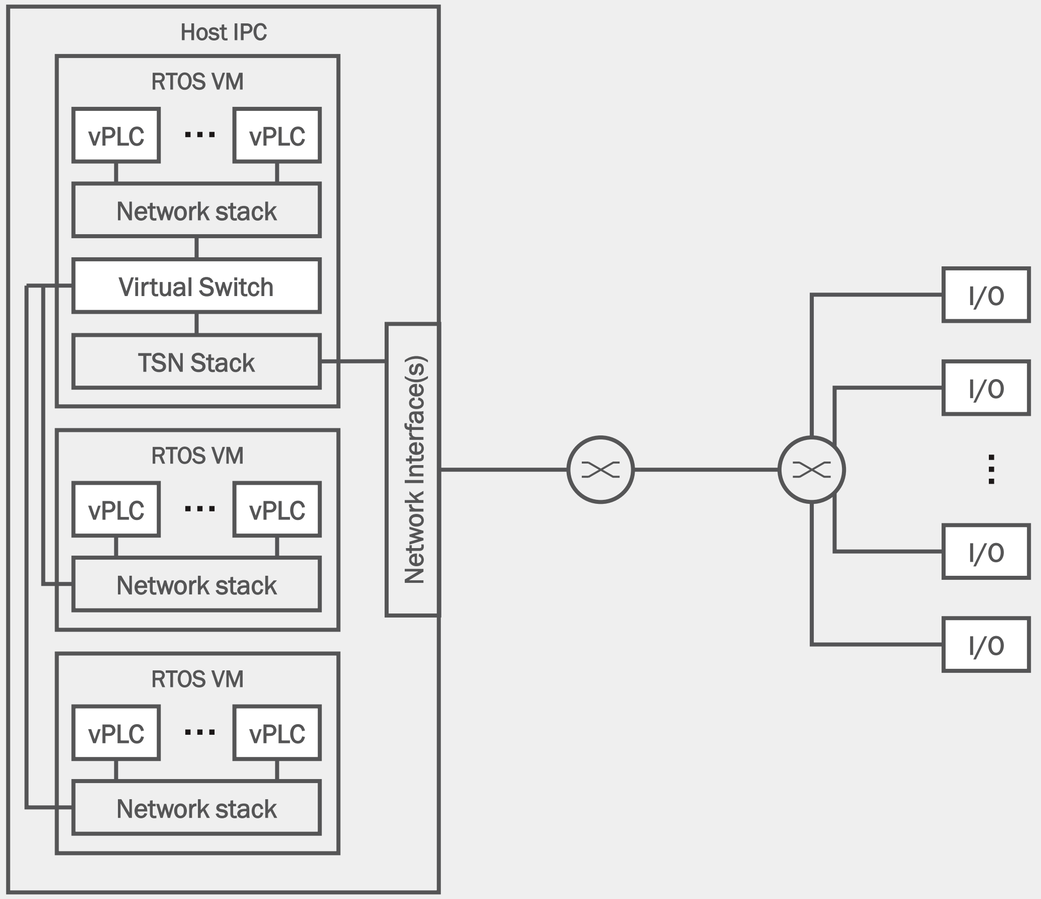
Practical Implementation Example: Prototype

- System hosts multiple RTOS VMs to consolidate PLC workloads
- Each VM routes network traffic through the Virtual Switch
- Scaling limits are determined by two factors
  - Resources required by each VM
  - Resources required by Virtual Switch



Practical Implementation Example: Prototype

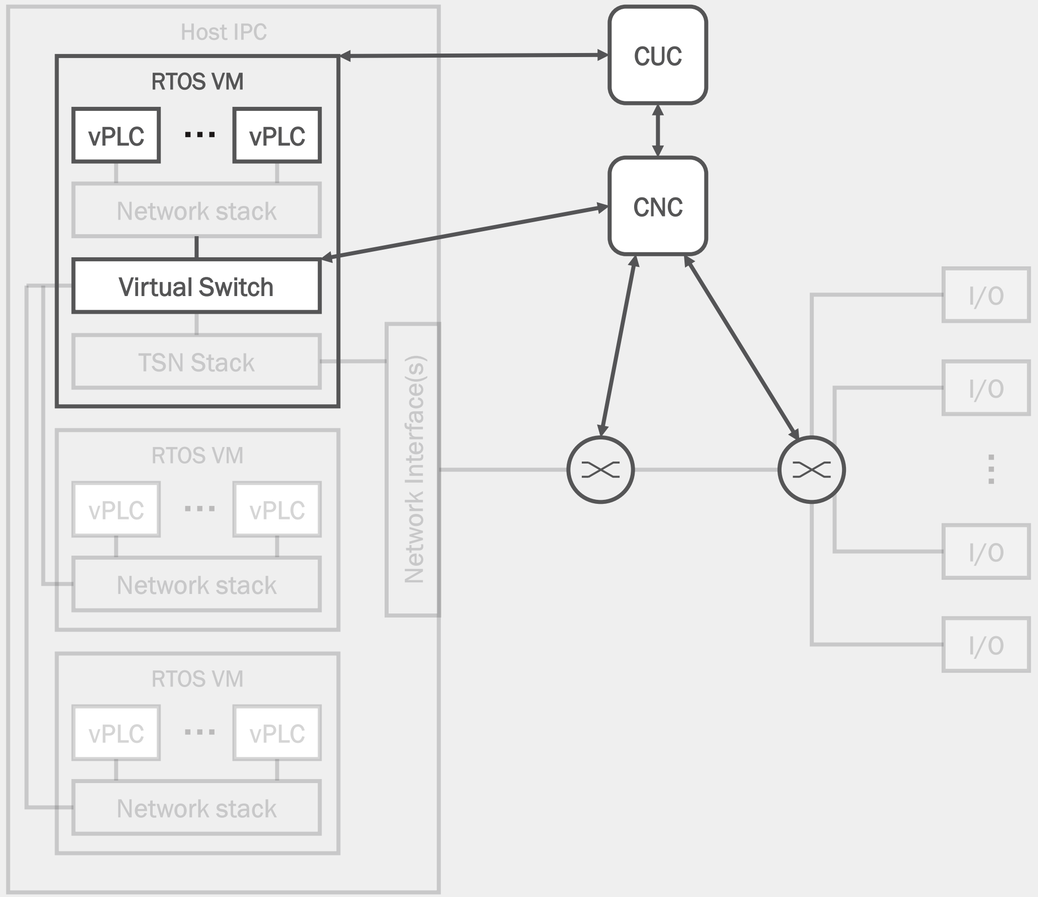
- Host interfaces to a TSN network
- TSN network hosts TSN-connected I/Os



Practical Implementation Example: Prototype

- Each VM is a TSN End Station configured by CUC
- Virtual and physical TSN switches require configuration from CNC

- Practical example – see workshop session at 16.00
  “Configuring TSN for Smart Factory with vPLC”



Initial Performance Data

Platform: Intel® Elkhart Lake with TCC enabled, i210 NIC, limited H/W offload.

Software: commercial vPLC for INtime RTOS.

Application: PLC engine + Profinet stack through TSN to simple I/Os.

A “VM” uses one physical core.

No virtual switch in initial assessment (only one VM).

One instance of the vPLC + TSN overhead consumes ~22% of one CPU core.

Adding another vPLC adds another ~18% CPU load.

Load dependent on application complexity.

This allows scaling to a few (~5 in this example) vPLC loads on one VM.

Next Steps

1. More validation. Cannot create such a system in isolation, need to make sure that the elements are conforming with appropriate standards and profiles – make use of sercices such as IIC testbed, Avnu Alliance.

2. Core platform, 13th-gen with TCC, i226 with more h/w offload.
   Virtual switch implementation with fixed configuration.
   ⇒ Many cores => many VMs, higher performance per VM.
   Measurement of TSN overhead vs. h/w offloading
   Where are the limits? Network bandwidth? CPU load? Memory?

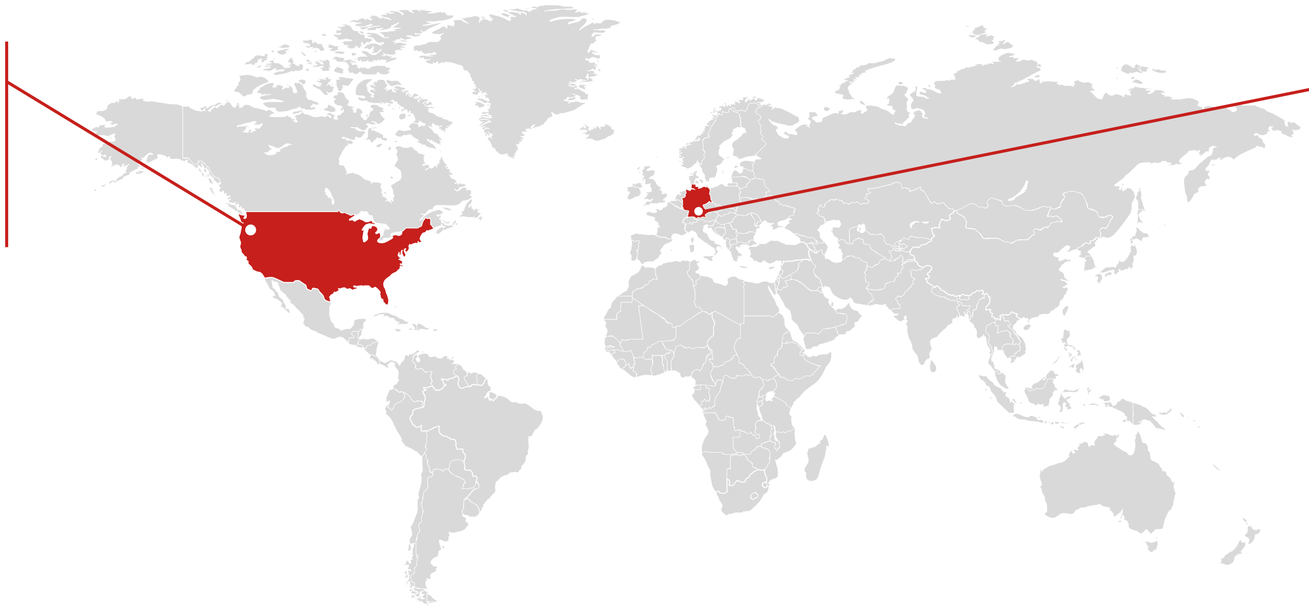
3. Full virtual switch implementation to TSN specs. Allows for more vPLCs up to network bandwidth.

4. How far can we scale up? Xeon platform, higher performance network interface?

Christopher Main

President & CTO, TenAsys

chris.main@tenasys.com | www.tenasys.com



TenAsys Corporation
1400 NE Compton Drive, Suite 301
Hillsboro, OR 97006
USA
+1 503 748 4720

TenAsys Europe GmbH
Hans-Stiessberger-Str. 2b
85540 Haar / Muenchen
Deutschland
+49 (89) 46 1498 45