New Communication Standard
Takyon Proposal Overview

November 2018
Khronos Exploratory Group Process

- **Proposal to create a new industry standard**
- **Exploratory Group** determines industry interest and creates a detailed Statement of Work (SOW)
- **Khronos Board** votes to establish a Working Group to execute the SOW
- **Working Group Meetings**

High-level discussions: industry need, requirements, design directions, identify possible future design contributions

Will there be Adoption of the proposed standard? Is Khronos the appropriate Standards Body to undertake and promote the work?

Detailed design work starts under Khronos IP Framework and Working Group Processes to build multi-company consensus

Multiple design contributions in addition to Takyon will be welcome if the working group is established

Invite industry feedback and input. Open to multiple possible future design contributions

*We Are Here!*
What Problem Does Takyon Solve?

The Problem

- TCP Socket
- UDP Socket plus multicast
- MPI
- MCAPI
- Verbs RDMA
- Network Direct RDMA
- GPIO
- Proprietary
- etc.

- GPU Direct
- GPU IPC
- GPU Memory

The Solution

- Process
- Thread
- Proprietary
- FPGA / IO Device

- Takyon
- GPU
- Thread

© Khronos® Group Inc. 2018 - Page 3
Summary of Proposed Solution

Message Passing API

Simple API Library
- Only 5 core functions
- Dynamic
- Fault tolerant
- Minimal overhead
- Optional “convenience” functions
- Implicit error checking option

Can be Layered Over Virtually any Interconnect e.g.
- Sockets
- POSIX mmap and shmem
- Local memcpy
- RDMA (Infiniband / RoCE / iWARP)
- GPUdirect / GPU IPC
- PCIe non-transparency

Reference Implementation Examples
- Simple use cases
- Performance characterization
- Fault tolerance
- MPI like “collective” examples
Where is it Relevant?

- Typical applications:
  - Radar / Radio Comms Processing
  - EO/IR video processing
  - Autonomous vehicles
  - Virtual reality / Augmented reality / Simulators
  - HPC computing
  - Industrial, Medical, IoT

Any “High Performance Computing” application where high speed, low latency, & deterministic data movement is critical.

Highly suited to heterogenous, SWaP constrained compute platforms.

- Reliable Point-to-Point
- Unreliable One-Sided (incl. multicast)
Reference Implementation

- **On Abaco’s GitHub**
  - [https://github.com/Abaco-Systems/axis-takyon](https://github.com/Abaco-Systems/axis-takyon)

- **Core API**
  - Core source code
  - Interconnects supported: Sockets (TCP, UDP datagram)
  - Memory copy (inter-process/thread)
  - x86 builds for Linux, Windows and MacOS

- **Extensions**
  - Additional “convenience” functions outside core API
  - “Collective” wrappers, timing utility functions, endian conversion

- **Examples**
  - Basic “hello world”, pipeline, performance, determinism, fault tolerance, IO, “collective”

- **Docs**
  - Quick reference and User’s Guide
Summary

The Takyon proposal aligns with rationale for a new communications standard

Key points

• Simple API
• Built for performance
• Built for fault tolerance
• Easily extendable
  - Wrappers for collective functions
  - Interconnect support
• Potential language support
  - C implementation
  - C++ bindings
  - Python bindings
Heterogenous Communications Exploratory Group

Takyon Proposal
Technical Overview
Point-to-Point Concept
Reliable message passing (data dropping not allowed)

- Connection “path” created between communicating threads
- Path provides bi-direction conduit for data movement
- Multi buffering built in (each direction can use different buffer counts and sizes)
One-Sided Concept
Unreliable message passing (data dropping allowed)

- Endpoint is either a sender or a receiver, but not both
- Multi buffering still allowed

Sender
- Great for streaming live output
- Multicast supported

Receiver
- Great for streaming live input
- Use with output devices: e.g. Lidar, Videos, GPS
Transfer Mechanism
Designed for modern interconnects

Zero Copy, One Way, Two Sided Transfers

- Zero Copy: Takyon’s knowledge of the message destination removes the need for additional copies
- One Way: Takyon’s knowledge of the message destination removes the need for back and forth coordination with the sender
- Two Sided: The receiver is notified when message arrives. No reliance on extra application messages or application-induced receive-side polling

Combining these three aspects are the key to achieving best latency, throughput, and determinism
Other Aspects to Performance & Determinism

- No implicit synchronization
  - Provides user with control
  - No hidden sync messages
  - Good for low-latency & determinism

- Choice of polling or event driven notification per path
  - Polling for lowest latency
  - Event driven for best resource utilization

- No dynamic registering / “pinning” of memory
  - Good for determinism
Dynamic & Fault Tolerant
Detecting and handling a bad path

The Mechanism
• Paths can be created and destroyed dynamically at application run-time
• Implicitly detect bad paths via timeouts
• Explicitly detect bad paths via return code

The Responsibility
• Takyon does not do any implicit fault recovery
• Application must use above mechanisms to build explicit fault recovery
Simple API
5 Core API Functions

Create a Path
TakyonPath *takyonCreate(TakyonPathAttributes *attributes)

Send Message
bool takyonSend(TakyonPath *path, int buffer_index, uint64_t bytes, uint64_t src_offset, uint64_t dest_offset, bool *timed_out_ret)

Test for Send Completion
bool takyonSendTest(TakyonPath *path, int buffer_index, bool *timed_out_ret)

Receive Message
bool takyonRecv(TakyonPath *path, int buffer_index, uint64_t *bytes_ret, uint64_t *offset_ret, bool *timed_out_ret)

Destroy a Path
char *takyonDestroy(TakyonPath **path_ret)

Key Benefit: Highly suitable for safety critical applications. Low API count good for ISO 26262 and DO-178B safety certification.
Suitable for safety-critical systems

- Validation of safety requirements simplified by small number of API functions
- Architecture of the implementation made robust by simplicity of mechanisms used by API
- Testability maximized by lack of implementation-defined, unspecified and undefined behavior
- Verification simplified by small state-space of the API
- Highest levels of safety certification, such ISO 26262 ASIL D and DO-178C DAL A, easy to achieve and maintain
API Extensions
Provided as reference source code to allow for easy customization

Endian
  takyonEndianIsBig()
takyonEndianSwapUInt16()
takyonEndianSwapUInt32()
takyonEndianSwapUInt64()

Time
  takyonSleep()
takyonTime()

Named Memory Allocations
  takyonMmapAlloc()
takyonMmapFree()

Path Attributes
  takyonAllocAttributes()

Collectives: Barrier
  takyonBarrierInit()
takyonBarrierRun()
takyonBarrierFinalize()

Collectives: Scatter
  takyonScatterSrcInit()
takyonScatterDestInit()
takyonScatterSend()
takyonScatterRecv()
takyonScatterSrcFinalize()
takyonScatterDestFinalize()

Collectives: Gather
  takyonGatherSrcInit()
takyonGatherDestInit()
takyonGatherSend()
takyonGatherRecv()
takyonGatherSrcFinalize()
takyonGatherDestFinalize()

Collectives: Reduce
  takyonReduceInit()
takyonReduceRoot()
takyonReduceChild()
takyonReduceFinalize()

Graph functions to define and load dataflow layouts from a configuration file
  takyonLoadGraphDescription()
takyonFreeGraphDescription()
takyonCreateGraphPaths()
takyonDestroyGraphPaths()
takyonPrintGraph()

Plus more

Plus more
IO Devices

API enables one-sided communication to IO devices

- From working example using automotive grade Lidar (Velodyne)

```c
// Create one sided Takyon path
TakyonPathAttributes attrs;
...
attrs.interconnect = "SocketDatagram -unicastRecv -server Any -port 2368"
TakyonPath *path = takyonCreate(&attrs);
// Get next raw UDP packet for lidar point cloud
uint64_t bytes_received;
takyonRecv(takyon_path, buffer_index, &bytes_received, NULL, NULL);
// Clean up
takyonDestroy(&takyon_path);
```

- Much simpler than alternative UDP socket code
Interop With Other Khronos Standards
Similar to how sockets are integrated with various Khronos standards

• EGL (Socket)
  1. Enable the socket interop via the extension: “EGL_NV_stream_socket_unix”
  2. Two remote applications establish a pair of connected sockets
  3. Each application passes its socket file descriptor to EGL function ‘eglCreateStreamKHR’
  4. The following flags are used to define the high level socket properties:
     - EGL_STREAM_PROTOCOL_SOCKET_NV
     - EGL_SOCKET_HANDLE_NV
     - EGL_SOCKET_TYPE_NV: EGL_SOCKET_TYPE_UNIX_NV, EGL_SOCKET_TYPE_INET_NV
  5. The mentioned function returns handle to a newly created object in each applications domain
  6. The objects communicate between each other over the socket connection

• EGL (Takyon Concept)
  1. Enable the Takyon interop via the extension: “EGL_NV_stream_takyon”
  2. Two remote applications establish a pair of connected sockets
  3. Each application passes its Takyon handle to EGL function ‘eglCreateTakyonStreamKHR’
  4. The interconnect type and properties could be extracted from the Takyon handle
  5. The mentioned function returns handle to a newly created object in each applications domain
  6. The objects communicate between each other over the socket connection

Similar interop can be achieved with other Khronos APIs
Summary

The Takyon API proposal was developed to address the need for a new standard for heterogeneous communications.

API simplicity

- Easy to learn
- Highly suitable for safety-critical applications - low API count good for ISO 26262 and DO-178B safety certification

API design

- Facilitates high performance implementations
- Facilitates determinism & fault tolerance
- Provides flexibility
  - Ability to layer on top most interconnects and IO devices
  - Extendibility for collective functions etc.
- Support for one-sided and two-sided communications
Links For Further Discussion

- Exploratory Website
  
  https://www.khronos.org/exploratory/heterogeneous-communication

- This website has links to:
  - A survey to gather feedback on your need for a new communication standard
  - A presentation discussing the rationale for a new communication standard