CCI

Common Communication Interface

Scott Atchley, David Dillow, Galen Shipman

Oak Ridge National Laboratory

Patrick Geoffray

Myricom

Jeffrey Squyres

Cisco Systems, Inc.

George Bosilca

University of Tennessee

Ronald Minnich

Sandia National Laboratories, Livermore

Applications are increasingly data-driven distributed services.

- These applications may control only one side of the pipe…
  - Common language: IP on the wire, Socket interface on the host.
  - Applications: web services, media delivery, trading exchange.
  - Not going away, way too much legacy.
- Or both sides of the pipe
  - No required wire protocol or programing interface.
  - Applications: back-ends, database, storage
    - Memcached, Big Table, Cassandra.
  - Socket interface hinders networking innovation.
  - Many vendor-specific interfaces available (dead or alive).

What if you control both sides ?

- Application developers either:
  - Stick with Sockets.
    - See substantially less benefit from current generation network technologies.
  - Lock themselves with a vendor-specific interface.
  - Support a number of different interfaces.
    - Requires deep expertise in multiple low-level network APIs
- Network vendors either:
  - Port Sockets on their low-level interface.
    - Limited performance.
  - Push their interface as the solution.
    - Everybody loves a good lock-in.
  - Support a number of different applications.
    - High support costs relative to potential revenue for niche applications.

Sockets

- Most widely used
  - Simple API
  - Robustness (failure tolerant)
  - Implicit buffering
  - Ubiquitous
- Unable to exploit many of the features of current-generation networking technologies
  - Cannot support zero-copy
  - Does not scale
    - In time: linear polling or interrupts.
    - In space: per socket resources.

MPI

- Designed as a bridge between application developers' and network vendors' needs in the High Performance Computing market
  - Standardization began nearly two decades ago
- MPI is the de-facto standard in HPC, Why not elsewhere?
  - High level of complexity
    - 200+ functions in MPI-1, 300+ in MPI-2
  - Original standard ignored dynamic environments
    - Added later but not widely adopted
  - Rigid fault model
    - Common fault case is abort execution of entire distributed application
    - Robust fault tolerance requires use of MPI dynamic process management (see above)

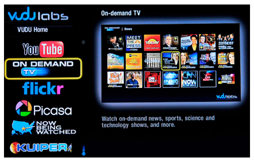
Specialized APIs abound

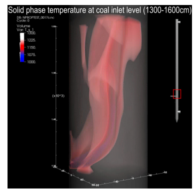
- OFA Verbs
  - High level of complexity, vendor lock-in is a concern
- Cray/Sandia's Portals
  - Highly specialized interface targeted towards HPC (MPI, SHMEM, UPC)
- Qlogics's PSM
  - Highly specialized interface targeted towards MPI
- Myricom's MX
  - Highly specialized interface targeted towards MPI
- IBM's LAPI and DCMF
  - Limited support outside of IBM network technologies
- DAPL
  - Limited support outside of iWARP capable devices
- LBL's GASnet
  - Designed specifically for the needs of UPC
- ARMCI
  - Designed specifically for the needs of Global Arrays
- LNET
  - Designed specifically for the needs of Lustre.
- BMI
  - Designed specifically for the needs of PVFS

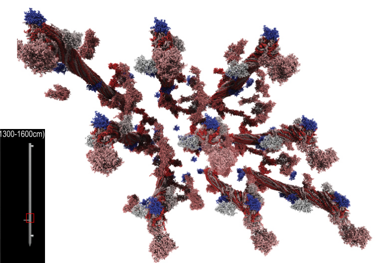
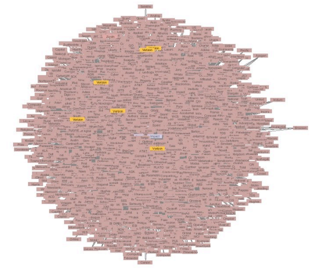
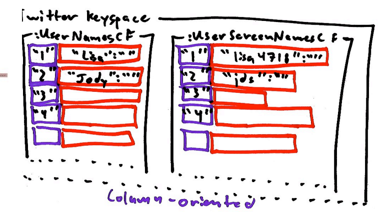
Summing up the landscape

	Sockets	MPI	Specialized APIs
Portability	✔	✔	✗
Simplicity	✔	✗	Varies
Performance	✗	✔	✔
Scalability	✗	✔	Varies
Robustness	✔	✗	Varies

Bridging two communities



Application Developers

Networking Technology
Vendors

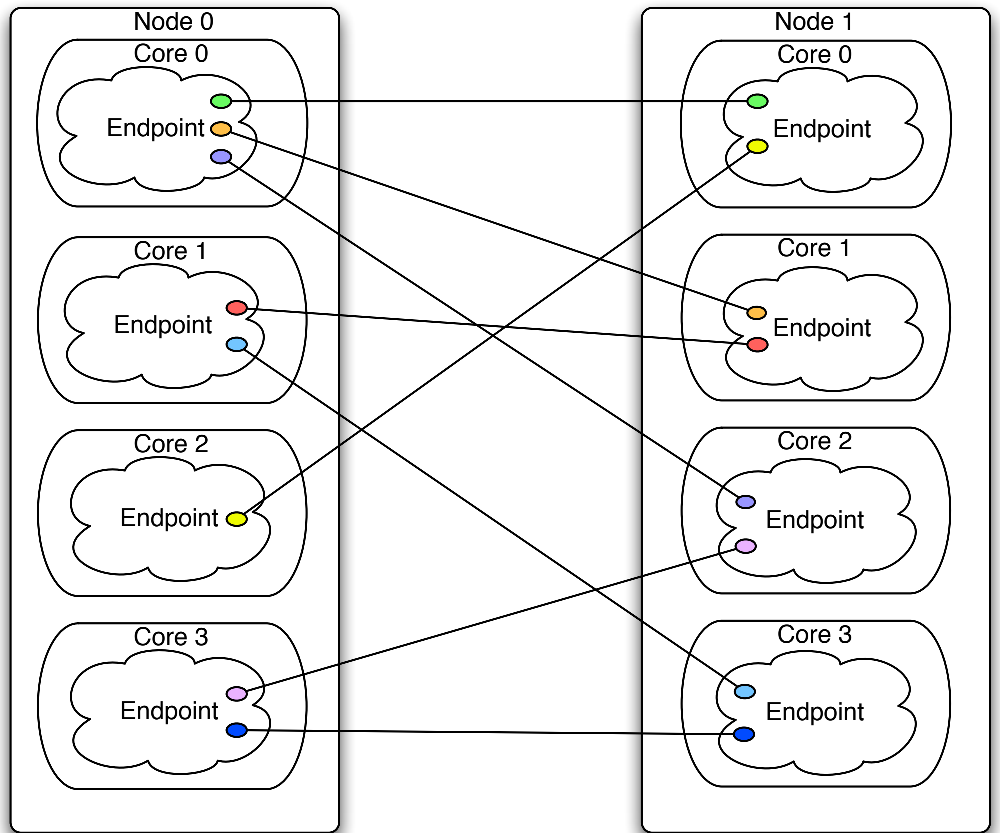



CCI design goals

- Performance
  - Can leverage OS-bypass, zero-copy, one-sided, async ops.
- Portability
  - Developers have limited resources.
  - Avoid vendor lock-in through a vendor neutral API.
- Simplicity
  - Must not be so complicated that only experts can use it.
  - Complexity tends to increase code size and maintenance cost.
- Scalability
  - Dynamic process management: peers come and go – not statically known a priori.
  - Time (polling) and space (buffer) cannot grow linearly with number of peers.
- Robustness
  - Need to contain faults to a single peer (i.e. fault isolation).

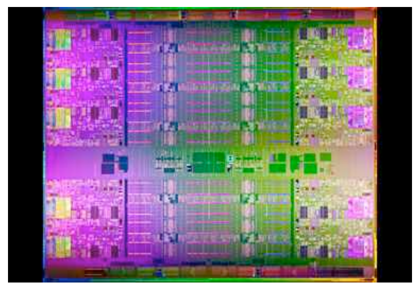
CCI Overview

- Endpoints
- Connections
- Communication
  - Active Messages
  - Remote Memory Access



CCI Endpoints

- Virtualized instance of a device – src/sink of communication
- Complete container of resources – queues and buffers
- An event driven model
  – Application may poll or block
  – Events include send, recv, connection establishment, etc.
  – Events may contain resources such as buffers
    - Resource ownership transfers to the application when the event is retrieved
  – May be returned out of order



Intel E7

CCI Connections

- Per peer - a single endpoint can handle many connections
- Scalable
  - no per-peer send/recv buffers
  - no per-peer event queues
- May have multiple connections to the same peer
- Use client/server connection model similar to Sockets
- Represents reliability and order attributes
  - Reliable with Ordered completion (RO)
  - Reliable with Unordered completion (RU)
  - Unreliable with Unordered completion (UU)
    - Multicast Send (MC_TX)
    - Multicast Receive (MC_RX)

Facebook data center

Active Messages

- Always buffered on both send and receive side
- Library manages buffers, not the application
- Events only, no handlers on receives
  - True handlers are the devil incarnate
  - Event includes pointer to data and the connection (peer)
- Message may be processed in-place
  - Even forwarded in-place
- May be copied out if needed long term
- Limited in size
  - Ideally MTU size to avoid segmenting/reassembly

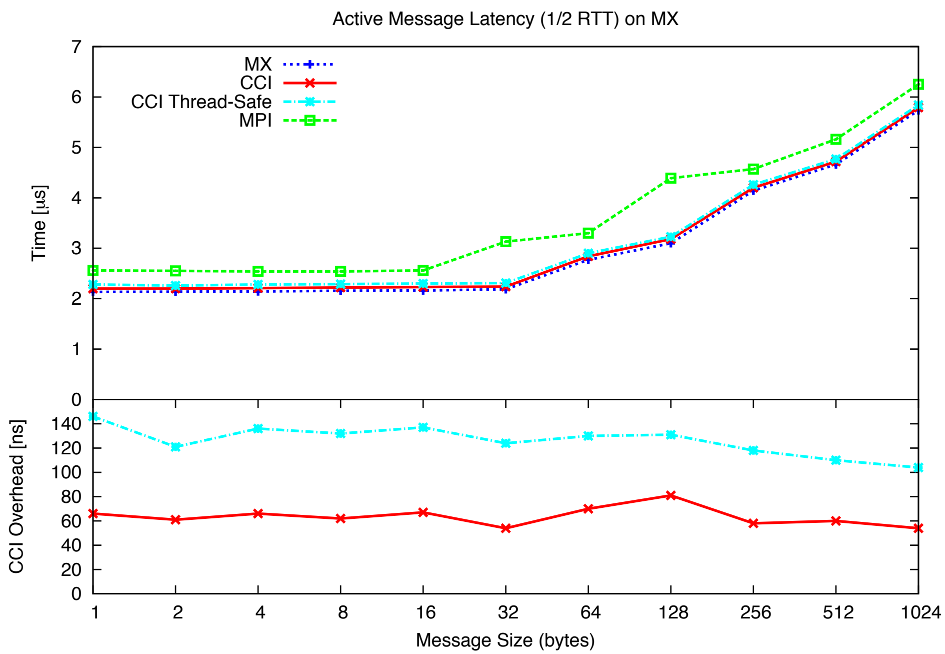
Remote Memory Access (RMA)

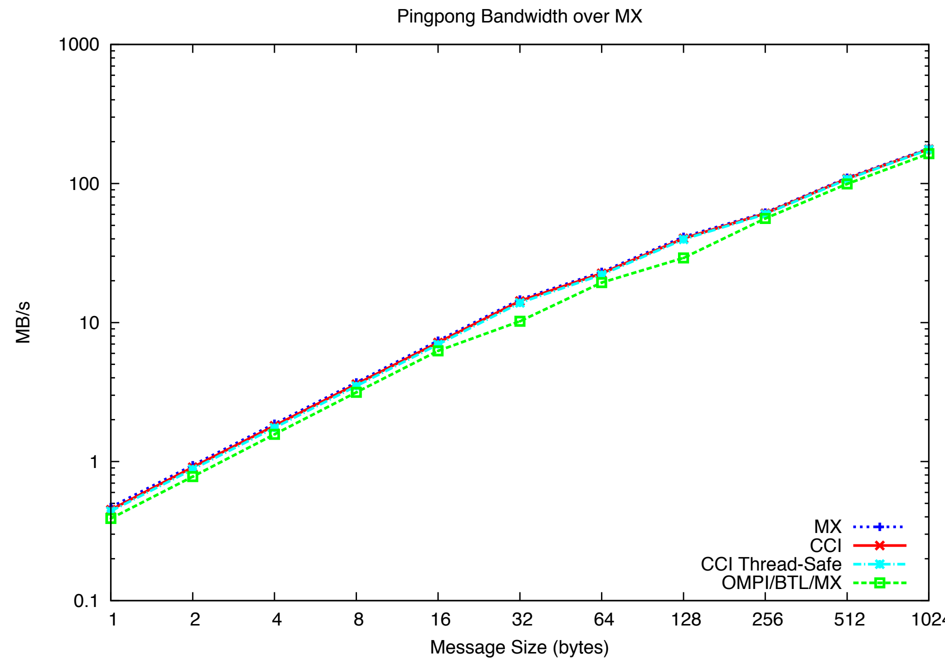
- RMA communication for bulk-data transfer
  - Zero-copy when available
  - One-sided operation
  - Active message model used for RMA synchronization
- Requires explicit memory registration
  - Provides broad portability
  - Simplified security model
- No intra-message or inter-message order guarantee
  - No last byte written last
- Optional inter-message ordering fence
- May be combined with immediate delivery of AM

Status and Evaluation

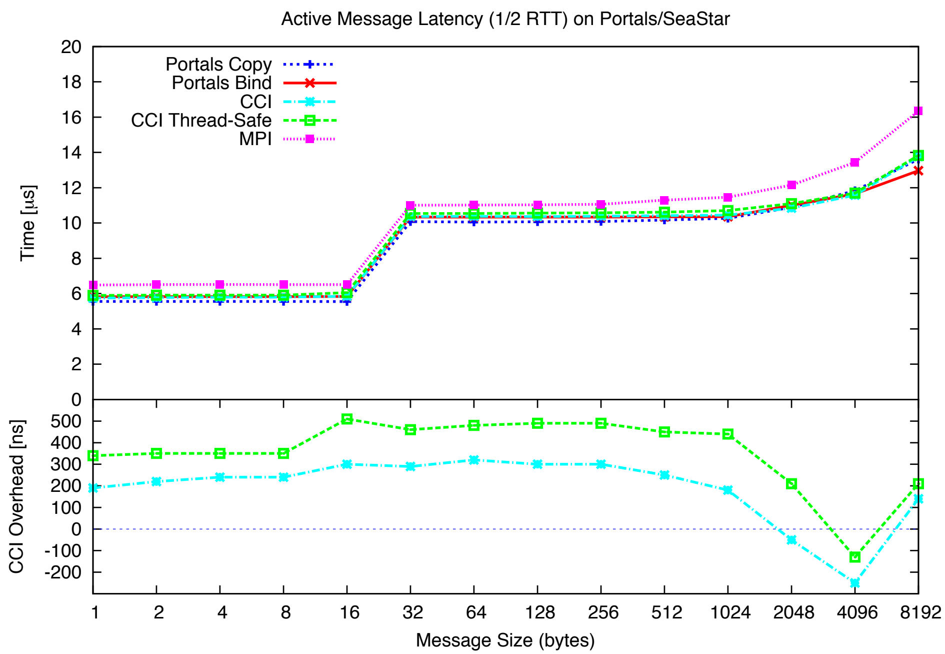
- ~~Three~~ Four proof-of-concept implementations
  - Sockets, MX, Portals 3.3, Native SeaStar
- Sockets
  - Uses UDP with one socket per endpoint, Implements reliability when required
  - Implements AM, RMA Write
- MX
  - Implements AM only
- Portals 3.3
  - Implements multiple endpoints using match bits
  - Implements AM, RMA Write, Read, and Fence
- Native SeaStar
  - Implements AM only
  - Working on adding RMA

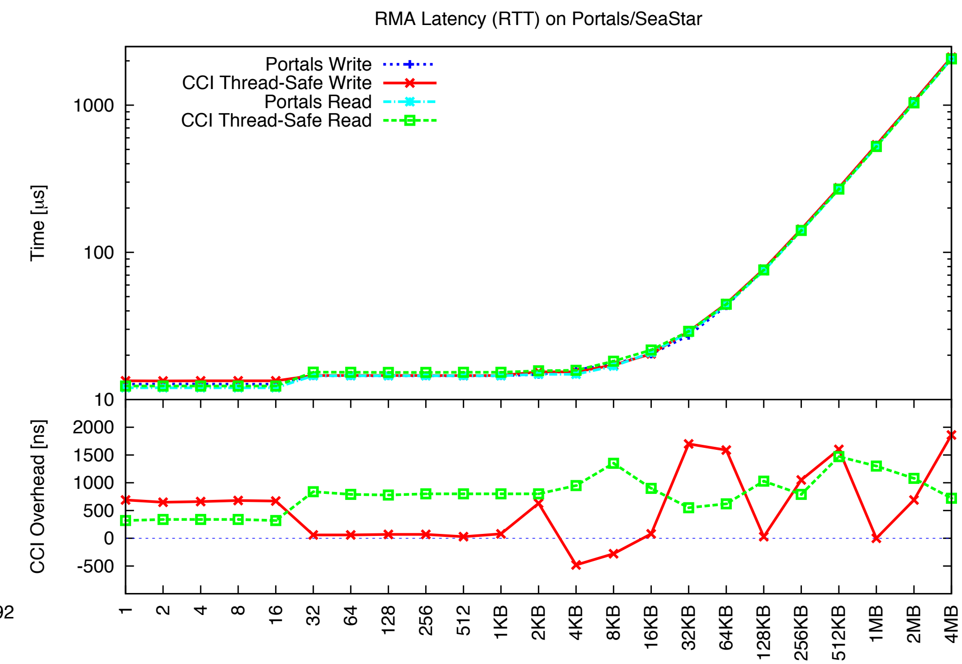
CCI/MX Performance



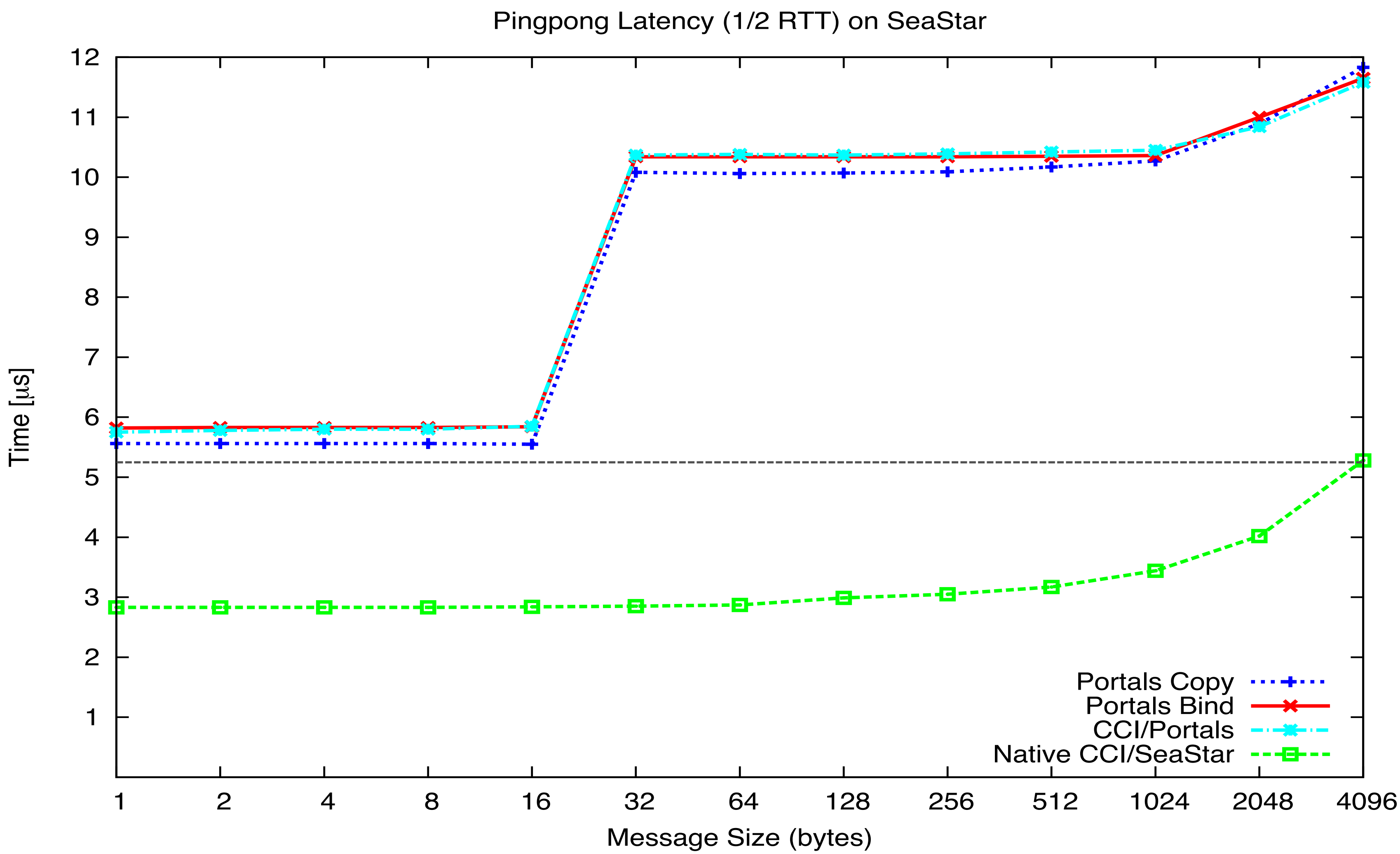


CCI/Portals Performance





Native SeaStar Performance



Caveats: Portals provides matching and thread-safety
Portals running on CNL, not Catamount
CCI/SS may require progress thread

Benefits of a common bridge

Application Developers

- Decrease complexity
- Port once, run everywhere
- Encourage competition among vendors
  - Fosters innovation
  - Improves cost effectiveness
- Mitigates technical and business risk of single vendor solution

Network Technology Vendors

- Increases total addressable market
  - Deliver performance to the masses
- Ability to expose innovation through a modern API
- Reduces costs
  - Eliminate per application support
  - Leverage community development of core API
  - Enables an ecosystem

Conclusion

- Distributed apps need
  - Performance - low latency, high throughput
  - To support transient peers and to isolate peer failures
  - To support large numbers of peers with bounded resources
  - Portable, simple programing interface
- CCI aims to satisfy these needs
  - Uses endpoints to bound time and space resources
  - Uses connections to provide peer fault isolation
  - Uses low-overhead active messages for small/control messages
  - Uses RMA for bulk movement and one-sided semantics
  - Provides good performance
  - Simple API
- CCI Next steps
  - Finish fleshing out TCP and native Portals implementations
  - Work is underway to provide Cray GNI, IBM Blue Gene, and InfiniBand Verbs support

Call for participation!

- We are a bunch of engineers
  - We don’t have a website
  - We don’t have a logo
  - We don’t have a glossy white-paper
  - But… We do have deep expertise in communication libraries
- We also have a community development model
  - Code is currently hosted on a private git-hub
  - License model is BSD/Apache style license
  - Contributor agreement is Apache style
- If you want to help contribute please contact us