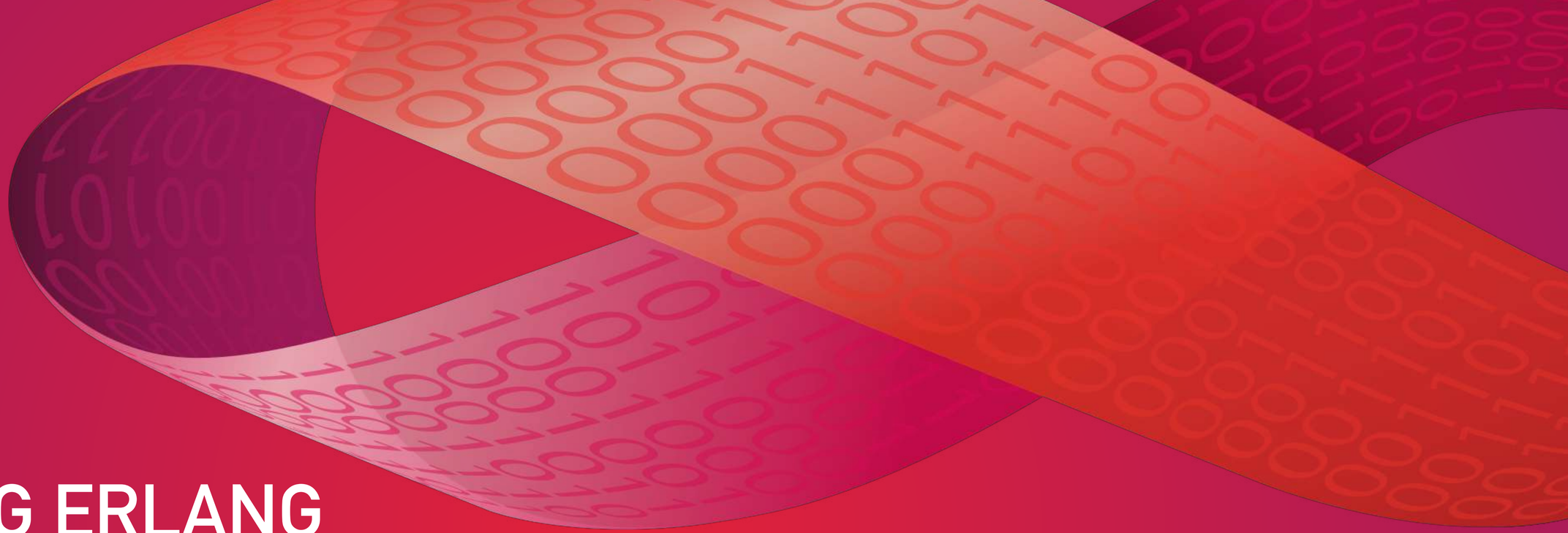


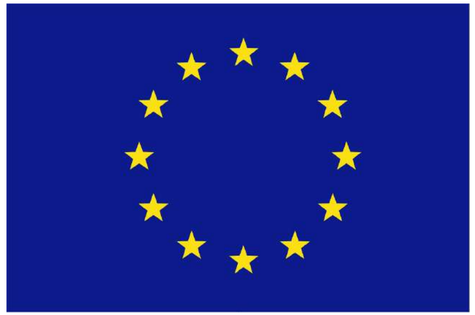
SCALING ERLANG
IN INDUSTRIAL PROJECTS

ERLANG DISTRIBUTION, UDP & TSN

DIPL. PHYS.
PEER STRITZINGER

LIGHTKONE

Lightweight computation for networks at the edge



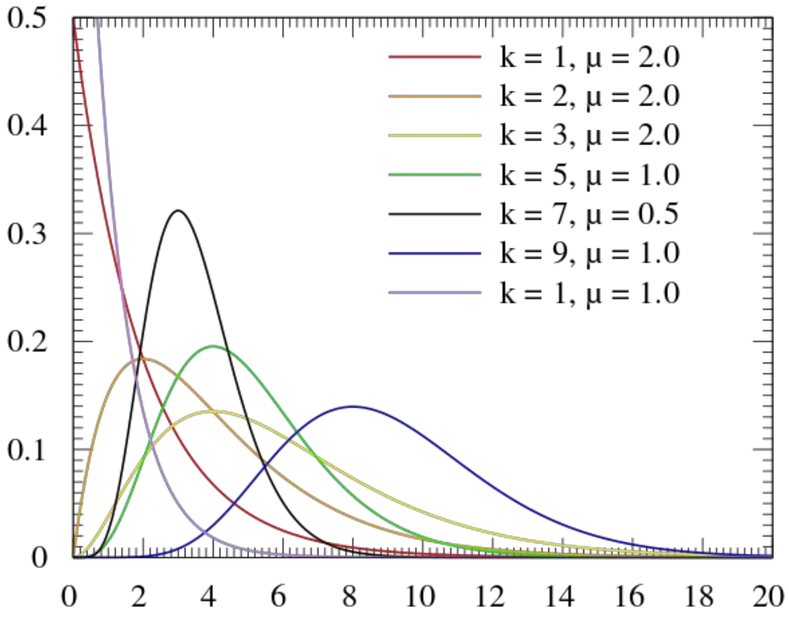
Funded by the Horizon 2020 Framework Programme of the European Union

- Computation at the Edge
- CRDTs
- Gossip protocols

DIPL. PHYS.
PEER STRITZINGER GMBH

ERLANG DISTRIBUTION

$$f(x;k,\mu) = \frac{x^{k-1}e^{-\frac{x}{\mu}}}{\mu^k(k-1)!} \quad \text{for } x,\mu \geq 0.$$



ERLANG DISTRIBUTION

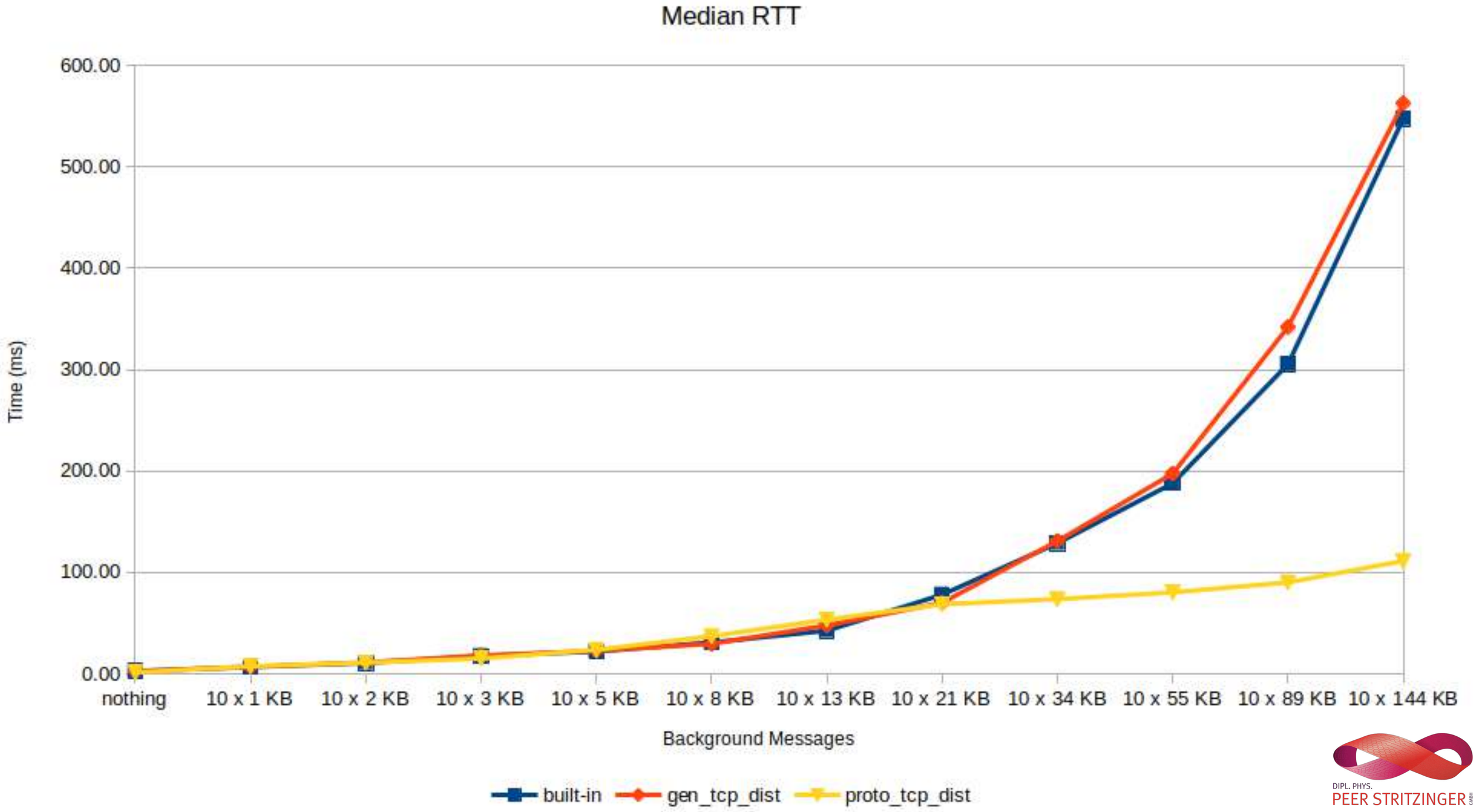
- Transparent distribution protocol
- Send Messages
- Link Processes
- Monitor Processes

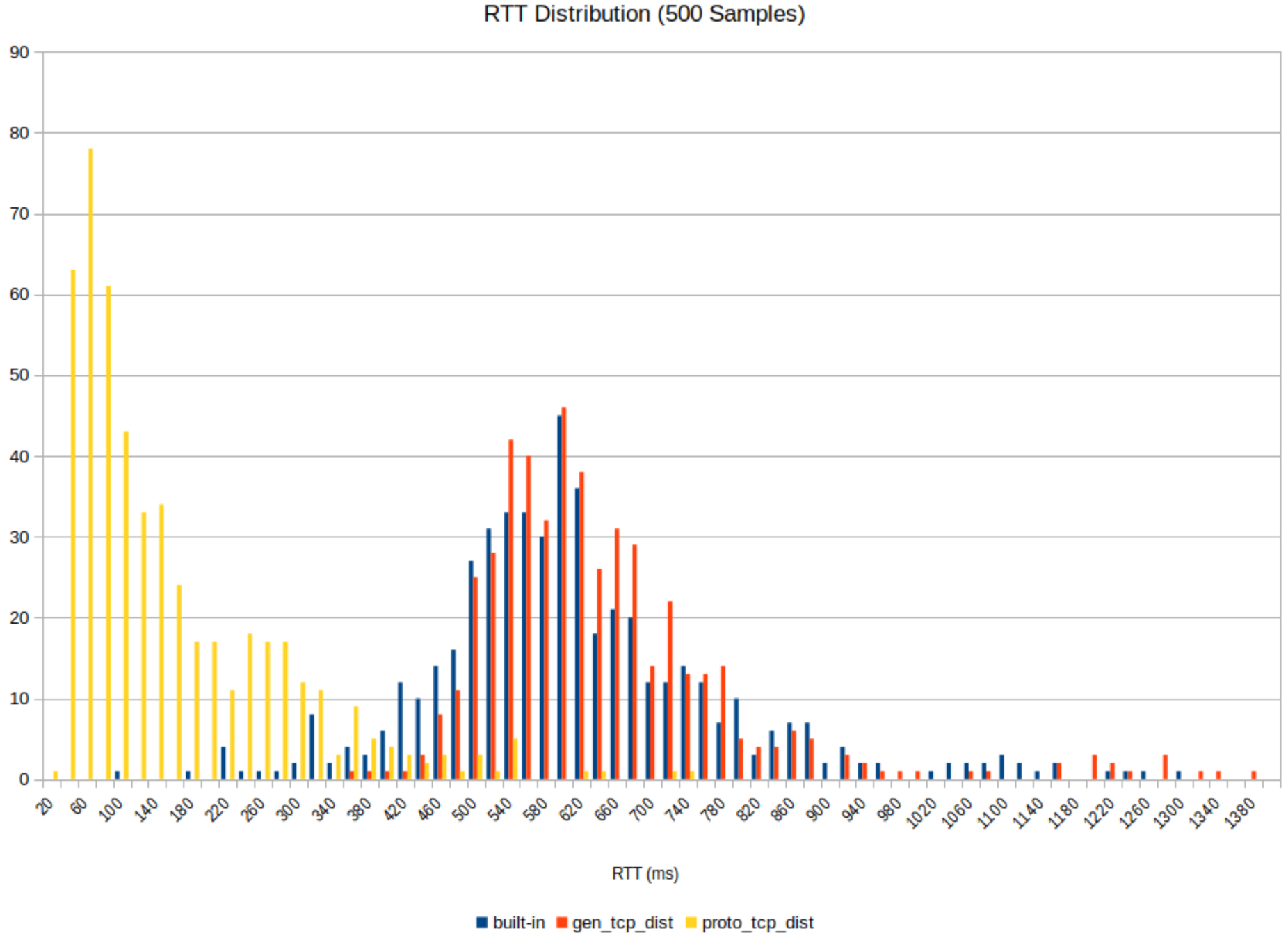
PROBLEMS

- Default is fully connected mesh
  - However there are hidden nodes
- Fully connected mesh doesn't scale well
- Global process registry requires fully connected mesh
- Head of line blocking

FIXING HEAD-OF-LINE BLOCKING

- `Channel = {From_pid, To_pid}`
- Round-robin scheduling
- Sequence number
- Message fragmentation
- Functionality like this will be in OTP 22





SEQUENCE NUMBERS AND FRAGMENTATION

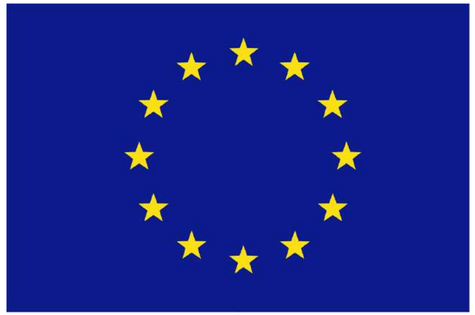
Why not use UDP?

UDP DISTRIBUTION PROTOTYPE

DIPL. PHYS.
PEER STRITZINGER GMBH

LIGHTKONE

Lightweight computation for networks at the edge



Funded by the Horizon 2020 Framework Programme of the European Union

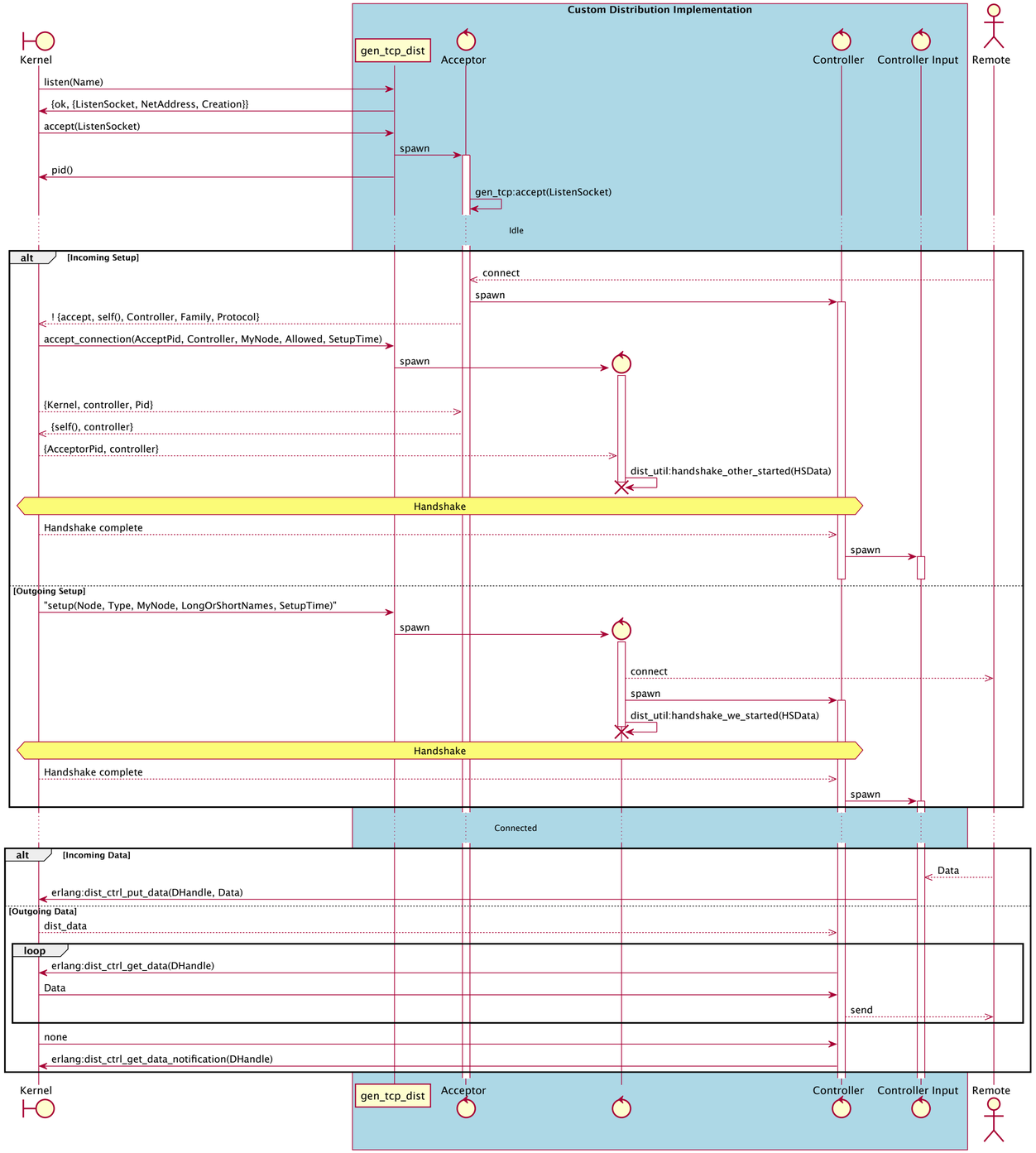
- Computation at the Edge
- CRDTs
- Gossip protocols

DIPL. PHYS.
PEER STRITZINGER GMBH

GOALS

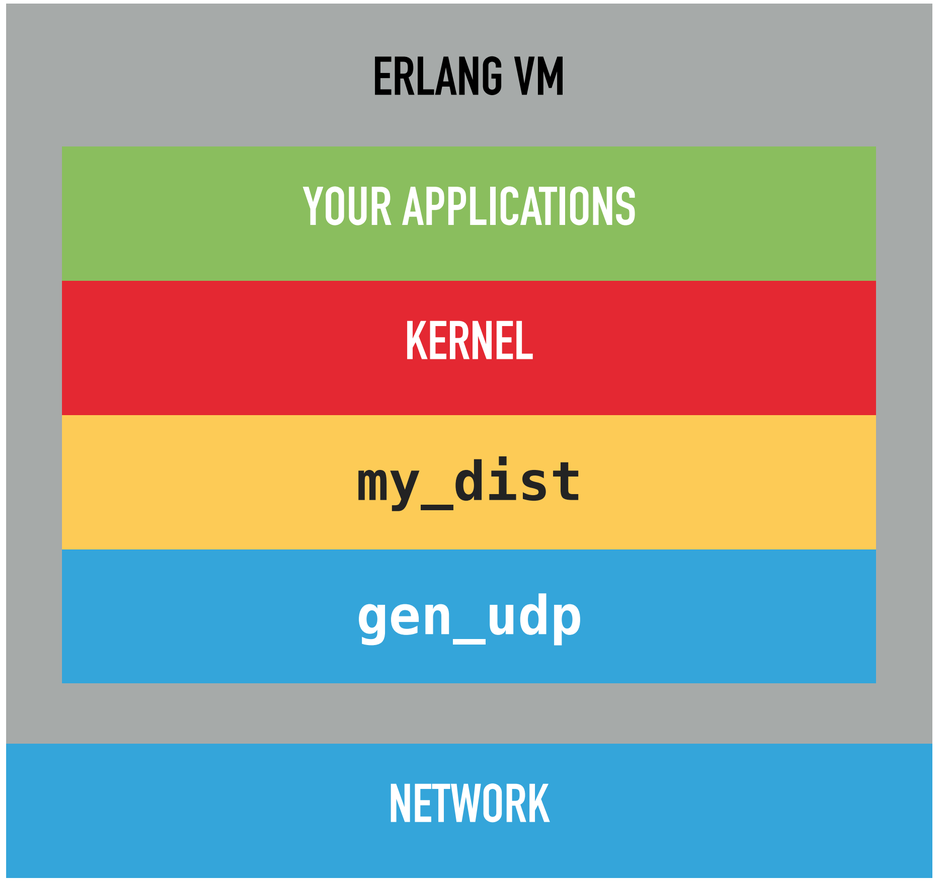
- Make UDP “work”
- Prototype ways of working around ahead-of-line blocking
- Discover challenges building custom distribution implementations

UH...

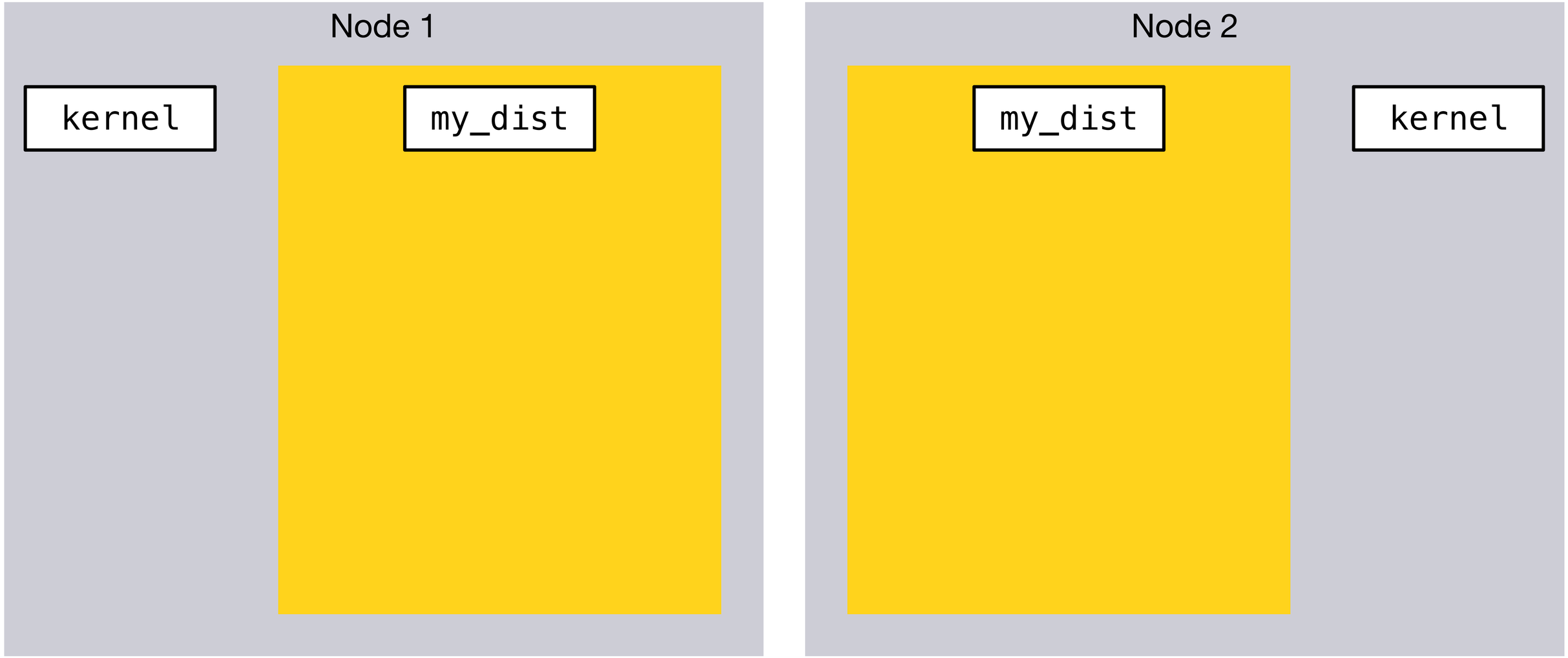


DIPL. PHYS.
PEER STRITZINGER GMBH

THE DISTRIBUTION “STACK”



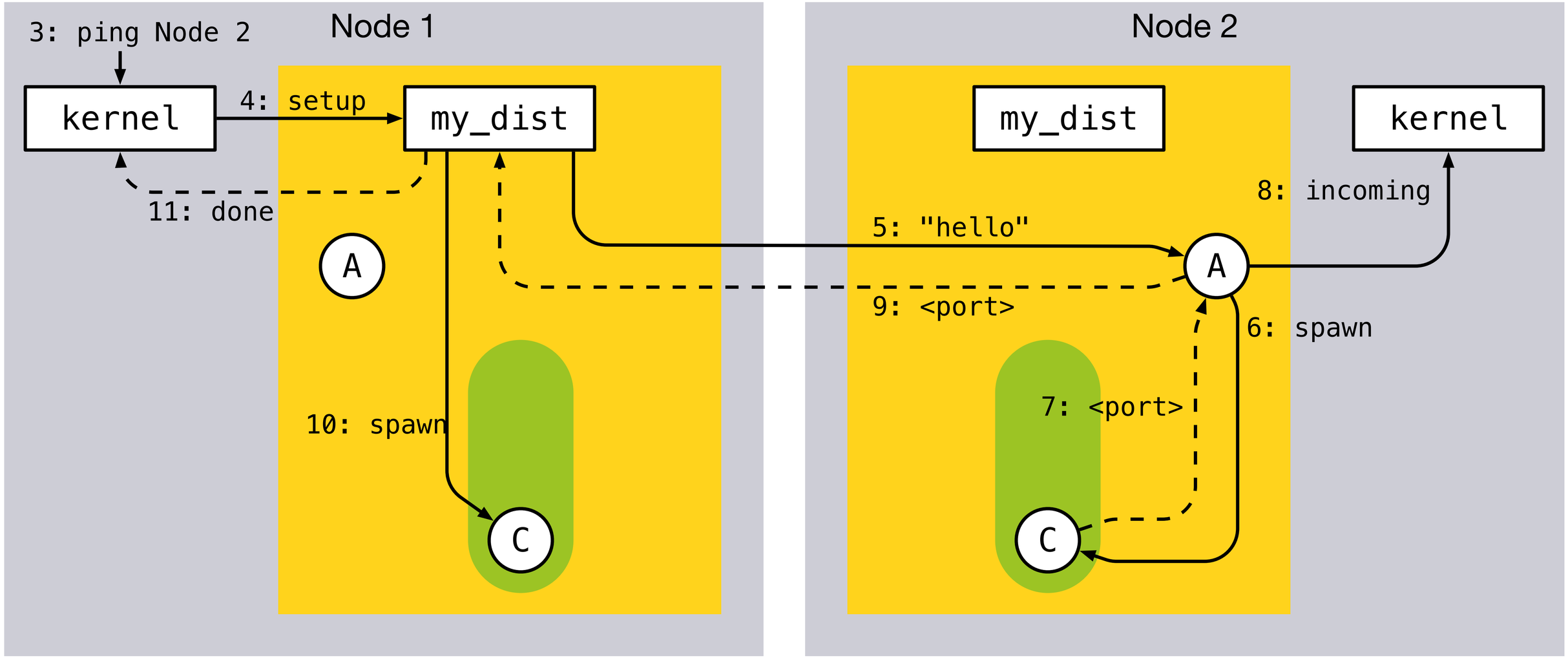
UDP DISTRIBUTION PROTOTYPE



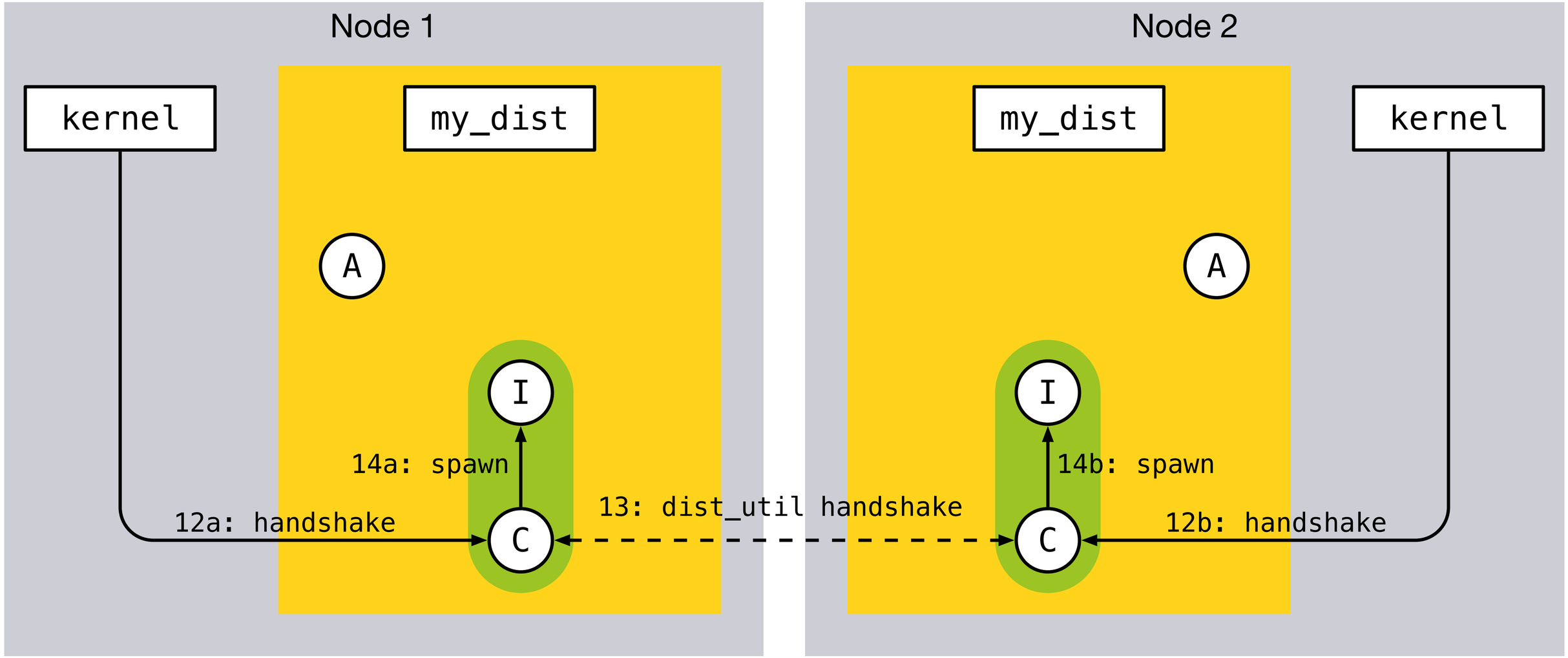
DIPL. PHYS.
PEER STRITZINGER

UDP DISTRIBUTION PROTOTYPE

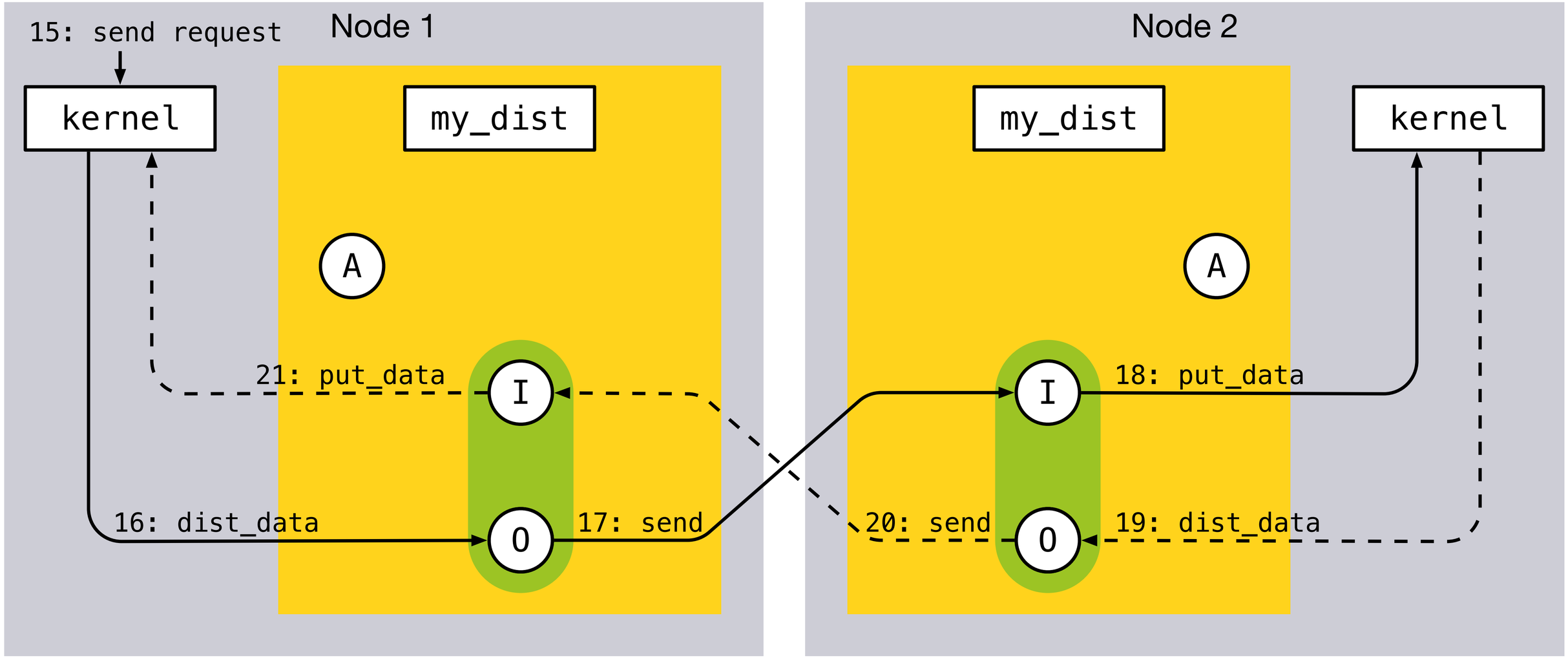
UDP DISTRIBUTION PROTOTYPE



UDP DISTRIBUTION PROTOTYPE



UDP DISTRIBUTION PROTOTYPE



UDP DISTRIBUTION PROTOTYPE SUMMARY

- One acceptor process per node
  - Opens a separate UDP listening port for connection attempts
- Two processes, one input and one output, per node connection
  - Could have been one process, but better throughput this way
- (not shown) Erlang Port Mapping Daemon (epmd) used to get the initial acceptor port to connect to

NEXT PROTOTYPE STEPS

CUSTOM DISTRIBUTION BEHAVIOR

DIPL. PHYS.
PEER STRITZINGER GMBH

MAKING CUSTOM DISTRIBUTION EASIER TO IMPLEMENT

- Help the developer by skipping the hard and complex parts
  - Complicated process model
  - Mixed API using both function calls, messages and callbacks
- We are investigating different levels of behaviors
  - High-level with different connection models
  - Low-level for abstraction out network handling only

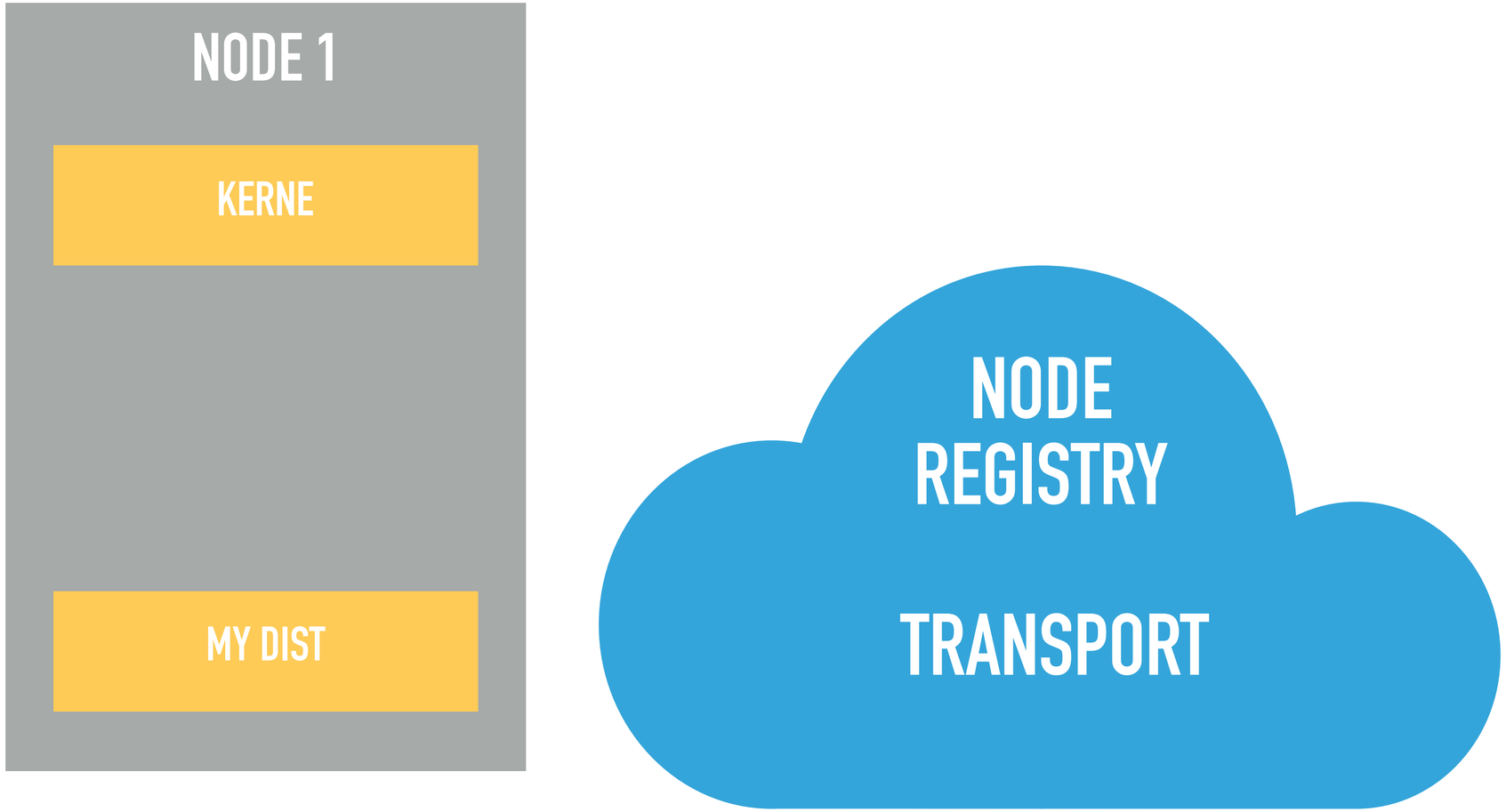
A DISTRIBUTION BEHAVIOR

- The UDP prototype is quite similar to the TCP example from OTP
- What if we could make a behavior that encapsulates all the complexity of the current API and process model?
  - What would such a behavior look like?
  - Are there other valid process models?
  - Can we combine this with a pluggable transport layer as well?

API PROPOSAL

- Acceptor
  - `acceptor_init/0`
  - `acceptor_info/2`
  - `acceptor_terminate/1`
- Controller
  - `controller_init/1`
  - `controller_send/2`
  - `controller_recv/1`
  - `controller_info/1`
  - Other socket related callbacks
    - Get/set opts, stats etc.

HIGH LEVEL APPROACH



TIME SENSITIVE NETWORKING

ETHERNET TSN

DIPL. PHYS.
PEER STRITZINGER GMBH

WHAT'S IT ABOUT

- Extensions to IEEE 802.1Q - Virtual LANs
- Low transmission latency
- High availability
- Converged networks:
  - Audio/Video Streams
  - Realtime Control systems

KEY COMPONENTS

- Time Synchronization
- Scheduling and traffic shaping
- Selection of communication paths
- Path reservations
- Fault tolerance

Time Synchronisation

IEEE802.1AS gPTP

IEEE802.1AS REV

Resource Management

IEEE802.1Qat SRP

IEEE802.1Qcc
SRP enhancement and performance improvement

Transport Stream and Control

IEEE1722 AVTP

IEEE1722.1 AVDECC

Scheduling

IEEE802.1Qav
FQTSS (CBS)

IEEE802.1Qch
Cyclic queueing and forwarding

IEEE802.1Qbv
Enhancements for Scheduled Traffic

IEEE802.1Qcr
Asynchronous Traffic Shaping

Preemption

IEEE802.1Qbu
Frame Preemption

IEEE802.1Qbr
Interspresing Express Traffic

Fault Tolerance

IEEE802.1CB
Frame Replication and Elimination for Reliability

IEEE802.1Qca
Path Control and reservation for redundancy

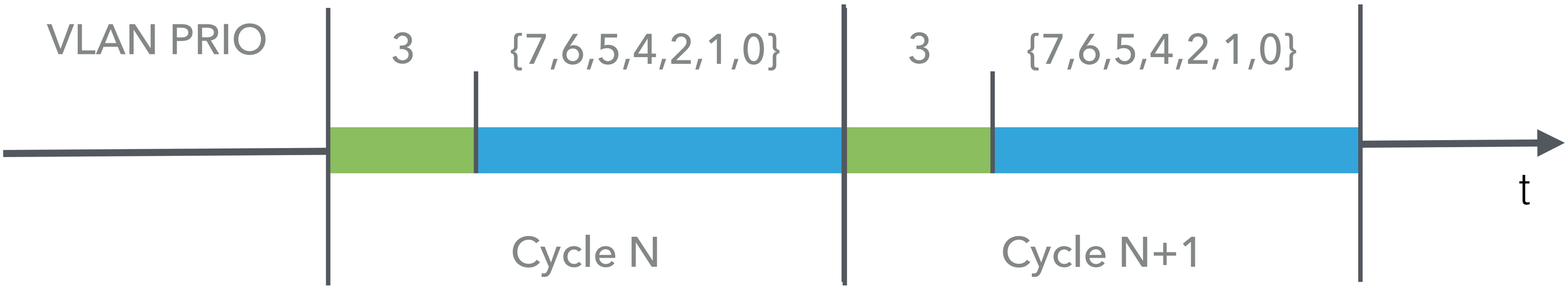
IEEE802.1Qci
Per Stream Filtering and Policing

AVB Protocol

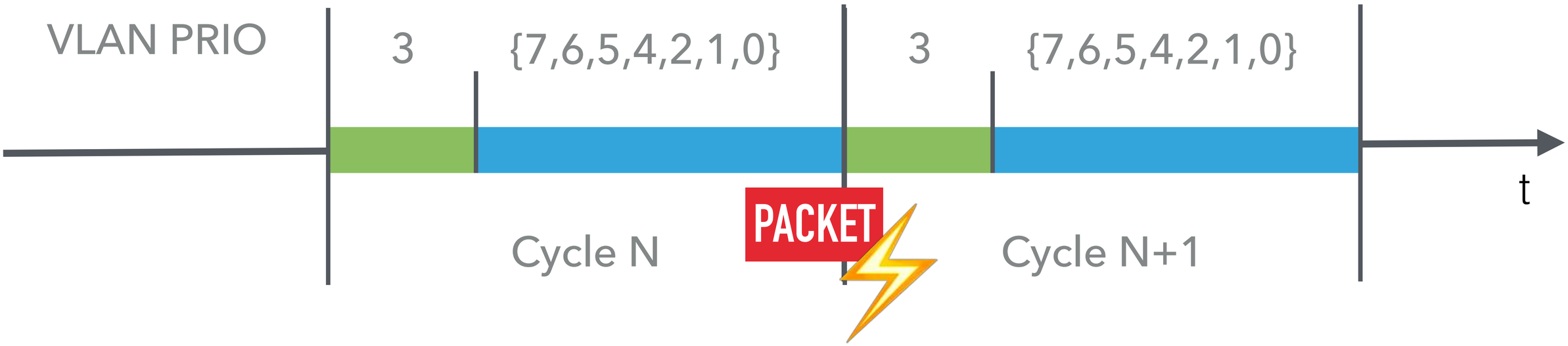
Newly Developed for TSN

DIPL. PHYS.
PEER STRITZINGER GMBH

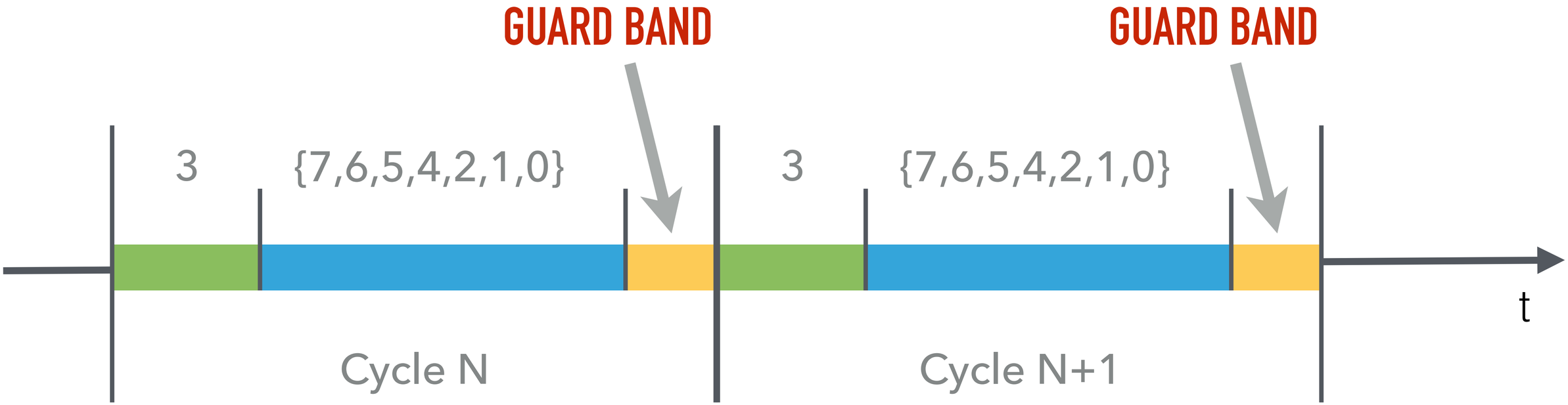
IEEE 802.1QBV



IEEE 802.1QBV



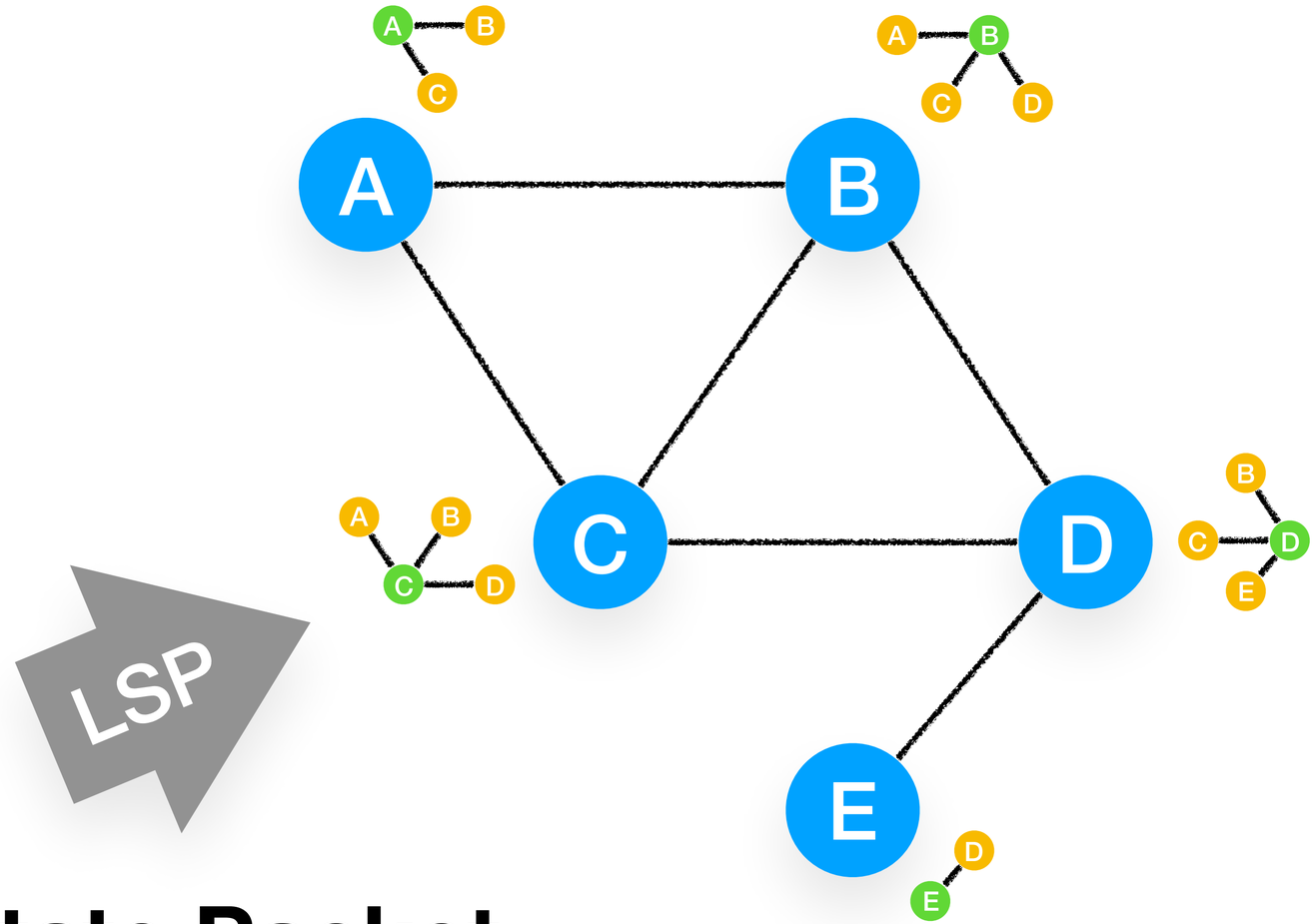
IEEE 802.1QBV



INTRO TO LINK STATE ROUTING PROTOCOLS

ROUTING AND DISCOVERY

DIPL. PHYS.
PEER STRITZINGER GMBH

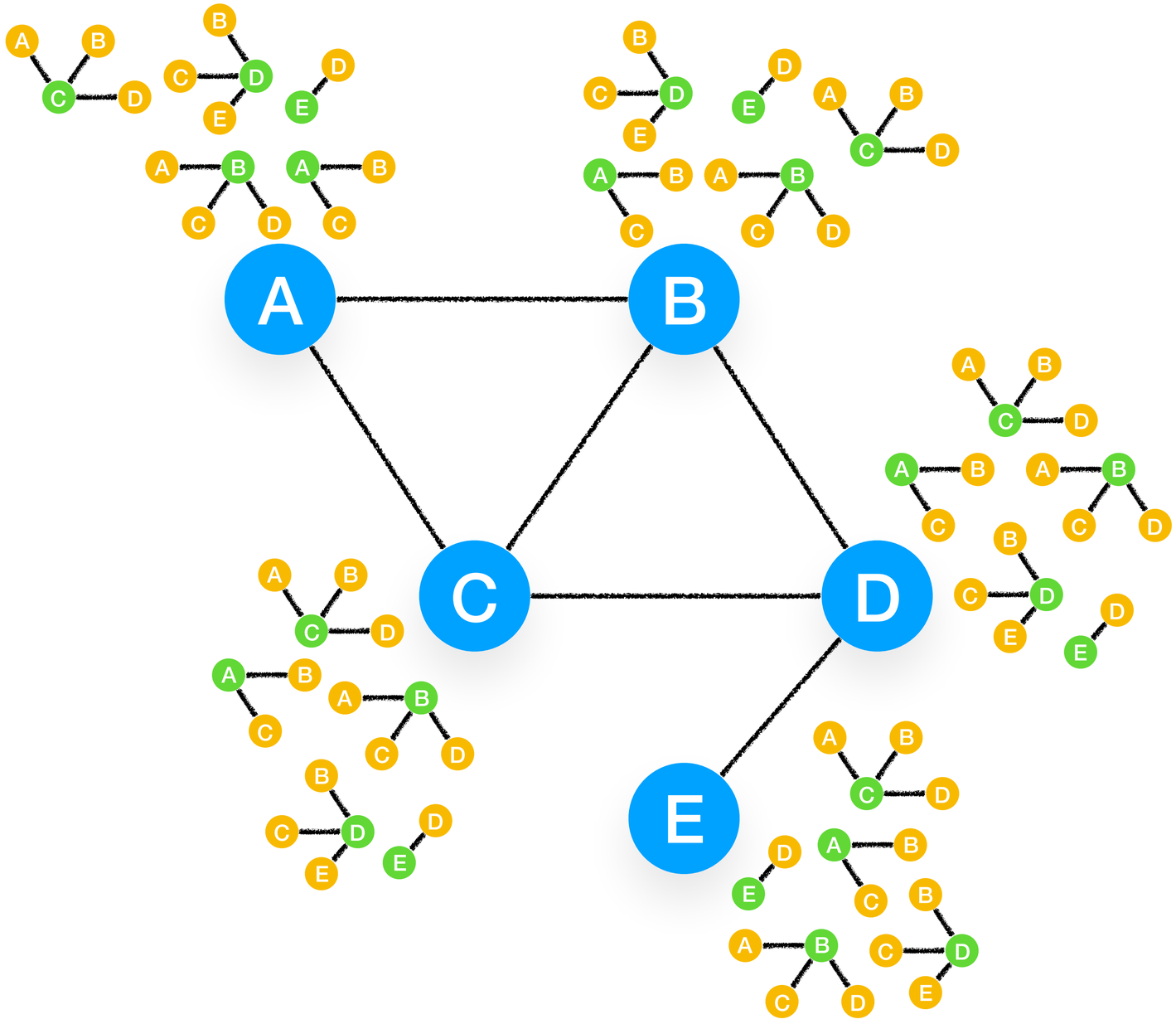


Link State Packet

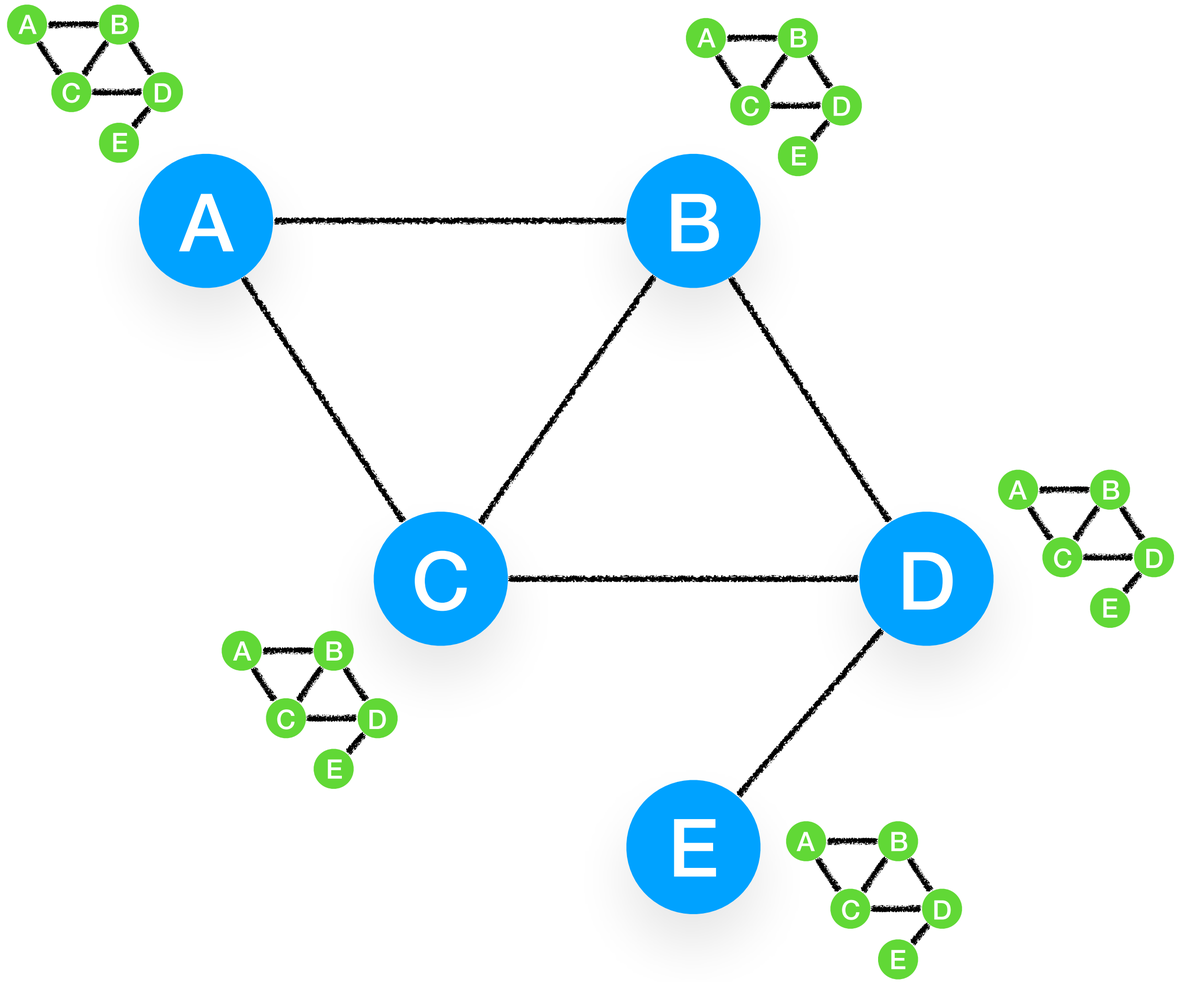
Magic Happens

RELIABLE FLOODING

DIPL. PHYS.
PEER STRITZINGER



DIPL. PHYS.
PEER STRITZINGER GMBH



DIPL. PHYS.
PEER STRITZINGER GMBH

IS-IS

- ISO/IEC 10589:2002
- Protocol agnostic
- De facto standard for large services provider network backbones
- IEEE 802.1aq Shortest Path Bridging

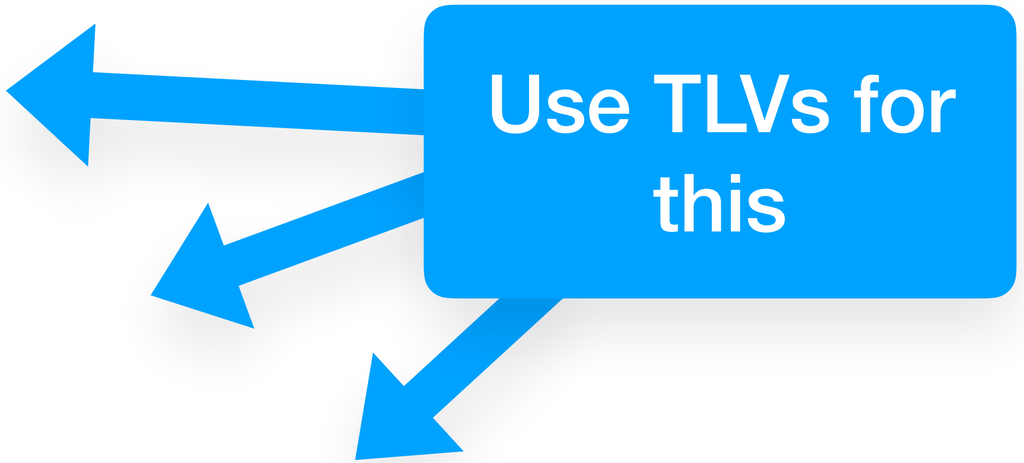
IS-IS CONT.

- Extensible
- Type Length Value (aka TLV) tuples
- 2 Layer Hierarchy

Attach your own arbitrary data labels to the network graph

WHAT CAN WE MAKE OUT OF THIS

- Node discovery
- Epmd replacement
- Global process registry
- Route Messages from Node to Node



SUMMARY

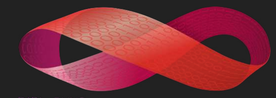
RESULTS & CURRENT STATE

- UPD prototype is working successfully
  - Proves it can be done
  - It does *not* handle packet loss
- TSN
  - Implementing a TSN Switch with Shortest-Path-Bridging
  - Using Erlang for the control plane

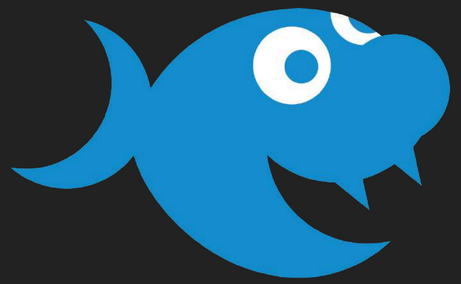
FUTURE WORK

- Erlang distribution
  - Connection oriented UDP?
  - QUIC
  - Virtual node connections
- Industrial networking
  - Prototype UDP over TSN
  - Implement real-time control prototypes

AND A SMALL ANNOUNCEMENT…



DIPL. PHYS.
PEER STRITZINGER



Back us now on KICKSTARTER

GRiSP 2

- Bare-metal Erlang
- Elixir & Nerves
- SoM module
- 696 Mhz - Faster CPU
- 128 Mb RAM - Twice the memory
- Wi-Fi *and* Ethernet

www.grisp.org

DIPL. PHYS.
PEER STRITZINGER

THANK YOU!

Questions?

www.stritzinger.com

www.grisp.org