Towards Exascale: Leveraging InfiniBand to accelerate the performance and scalability of Slurm jobstart.

Artem Y. Polyakov, Joshua S. Ladd, Boris I. Karasev

Nov 16, 2017

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™

Agenda

- **Problem description**
  - Slurm PMIx plugin status update
  - Motivation of this work
- **What is PMIx?**
  - RunTime Enviroment (RTE)
  - Process Management Interface (PMI)
  - PMIx endpoint exchange modes: full modex, direct modex, instant-on
- **PMIx plugin (Slurm 16.05)**
  - High level overview of a Slurm RPC
- **PMIx plugin (Slurm 17.11) – revamp of OOB channel**
  - Direct-connect feature
  - Revamp of PMIx plugin collectives
  - Early wireup feature
  - Performance results for Open MPI

Agenda

- **Problem description**
  - Slurm PMIx plugin status update
  - Motivation of this work
- **What is PMIx?**
  - RunTime Enviroment (RTE)
  - Process Management Interface (PMI)
  - PMIx endpoint exchange modes: full modex, direct modex, instant-on
- **PMIx plugin (Slurm 16.05)**
  - High level overview of a Slurm RPC
- **PMIx plugin (Slurm 17.11) – revamp of OOB channel**
  - Direct-connect feature
  - Revamp of PMIx plugin collectives
  - Early wireup feature
  - Performance results for Open MPI

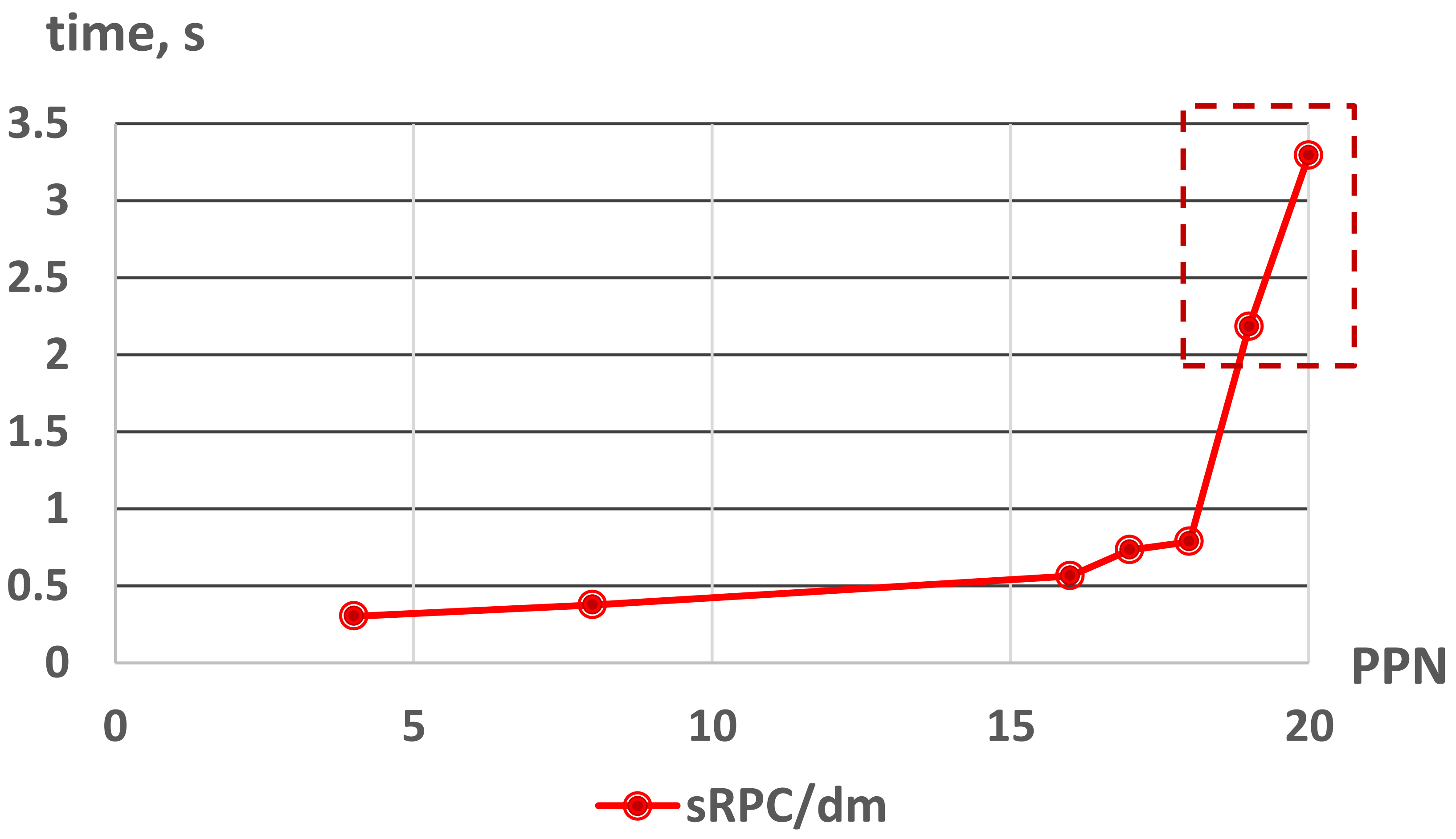
3089921ad9c0ad05372f955521b12f0c93ec73ec

Slurm PMIx plugin status update

- Slurm 16.05
  - PMIx plugin was provided by Mellanox in Oct, 2015 (commit 3089921)
  - Supports PMIx v1.x
  - **Uses Slurm RPC for Out Of Band (OOB) communication (derived from PMI2 plugin)**
- Slurm 17.02
  - Bugfixing & maintenance
- Slurm 17.11
  - Support for PMIx v2.x
  - **Support for TCP- and UCX-based communication infrastructure**
    - **UCX: `./configure … --with-ucx=<ucx-path>`**

Motivation of this work

- OpenSHMEM jobstart with Slurm PMIx/direct modex (explained below)
- Time to perform shmem_init() is measured
- Significant performance degradation when Process Per Node (PPN) count was reaching available number of cores.
- Profiling identified that the bottleneck is the communication subsystem based on Slurm RPC (sRPC).



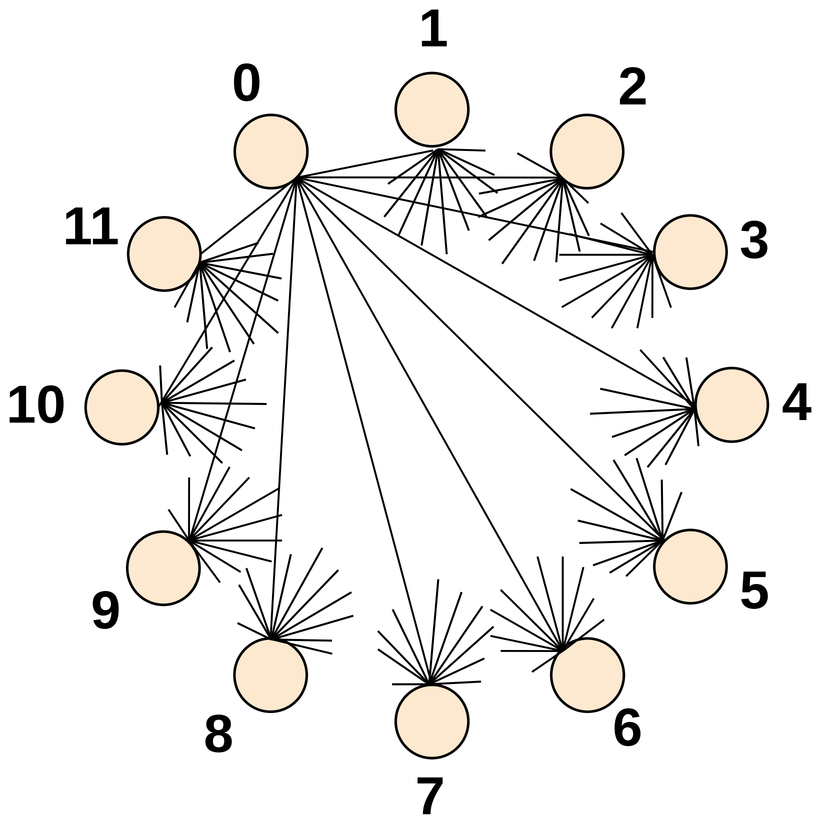
Measurements configuration:

- 32 nodes, 20 cores per node
- Varying PPN
- PMIx v1.2
- OMPI v2.x
- Slurm 16.05

Agenda

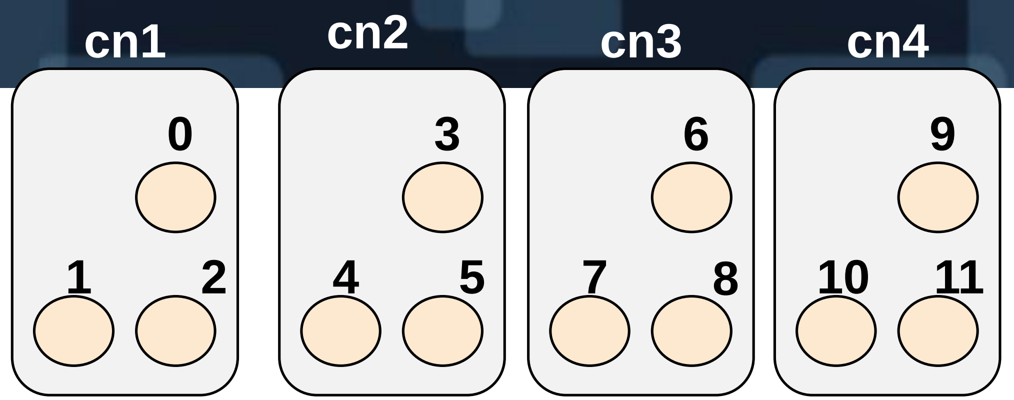
- **Problem description**
  - Slurm PMIx plugin status update
  - Motivation of this work
- **What is PMIx?**
  - RunTime Enviroment (RTE)
  - Process Management Interface (PMI)
  - PMIx endpoint exchange modes: full modex, direct modex, instant-on
- **PMIx plugin (Slurm 16.05)**
  - High level overview of a Slurm RPC
- **PMIx plugin (Slurm 17.11) – revamp of OOB channel**
  - Direct-connect feature
  - Revamp of PMIx plugin collectives
  - Early wireup feature
  - Performance results for Open MPI

RunTime Environment (RTE)



LOGICALLY

- MPI program: set of execution branches
- each uniquely identified by a *rank*
- fully-connected graph
- set of comm. primitives provide the way for ranks to exchange the data

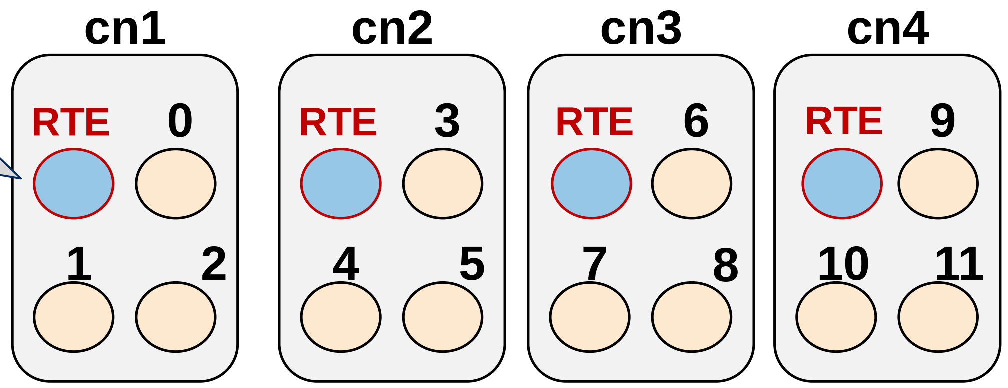


IMPLEMENTATION:

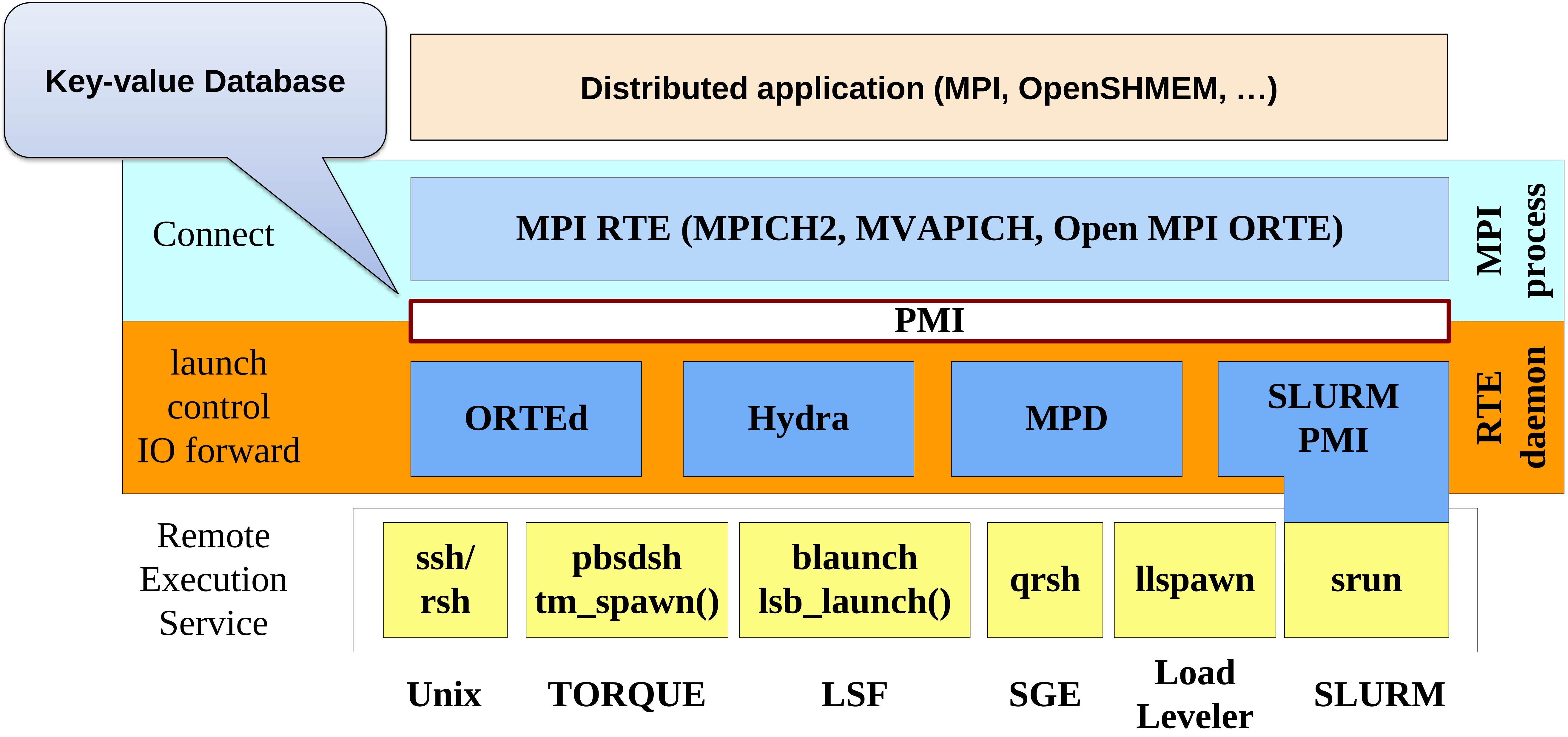
- execution branch = OS process
- full connectivity is not scalable
- execution branches are mapped to physical resources: nodes/sockets/cores.
- comm. subsystem is heterogeneous: intra-node & inter-node set of communication channels are different.
- OS processes need to be:
  - **launched**;
  - transparently **wired up** to enable necessary abstraction level;
  - **controlled** (I/O forward, kill, cleanup, prestage, etc.)
- Either MPI implementation or Resource Manager (RM) provides RTE process to address those issues.

topic of this talk

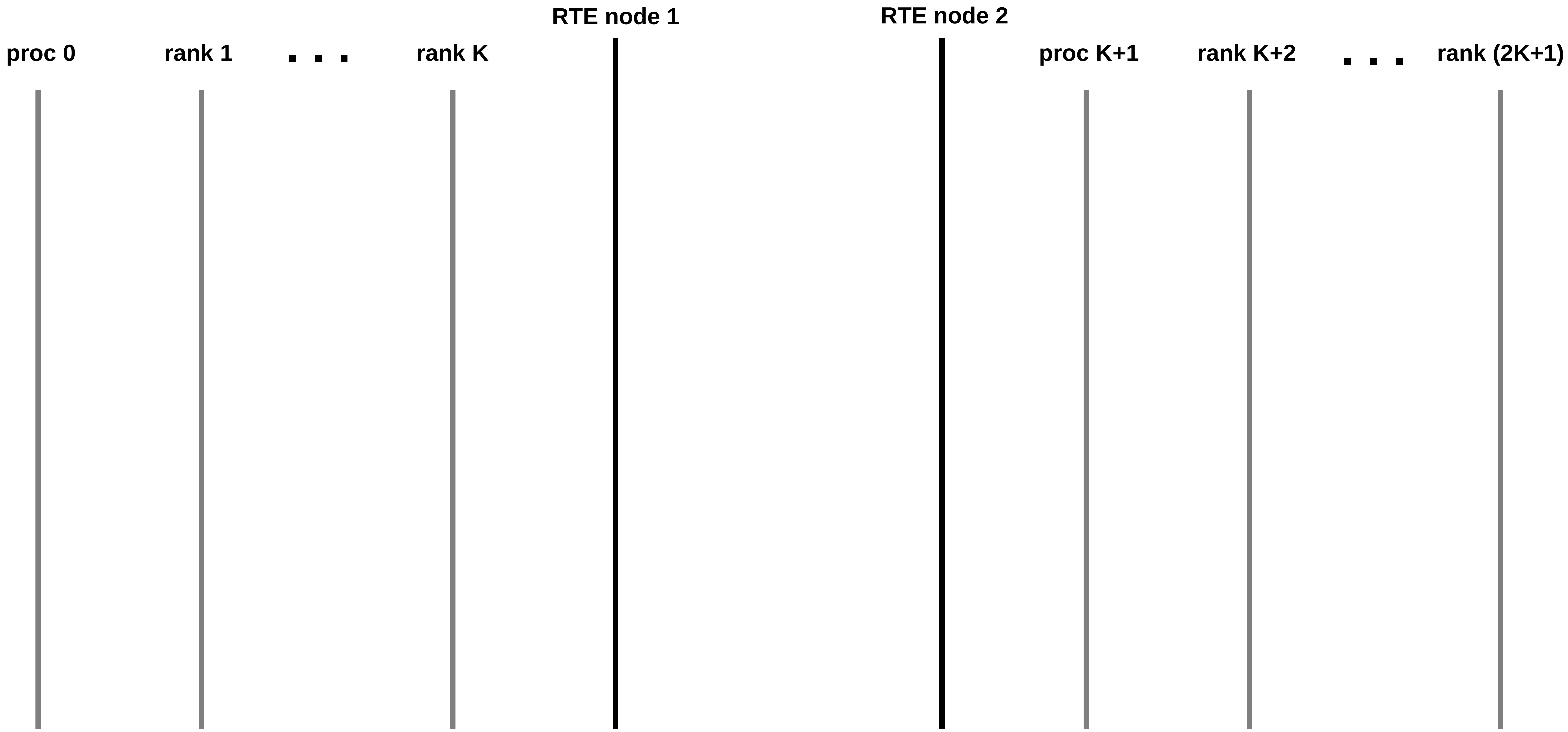
RTE daemon management process



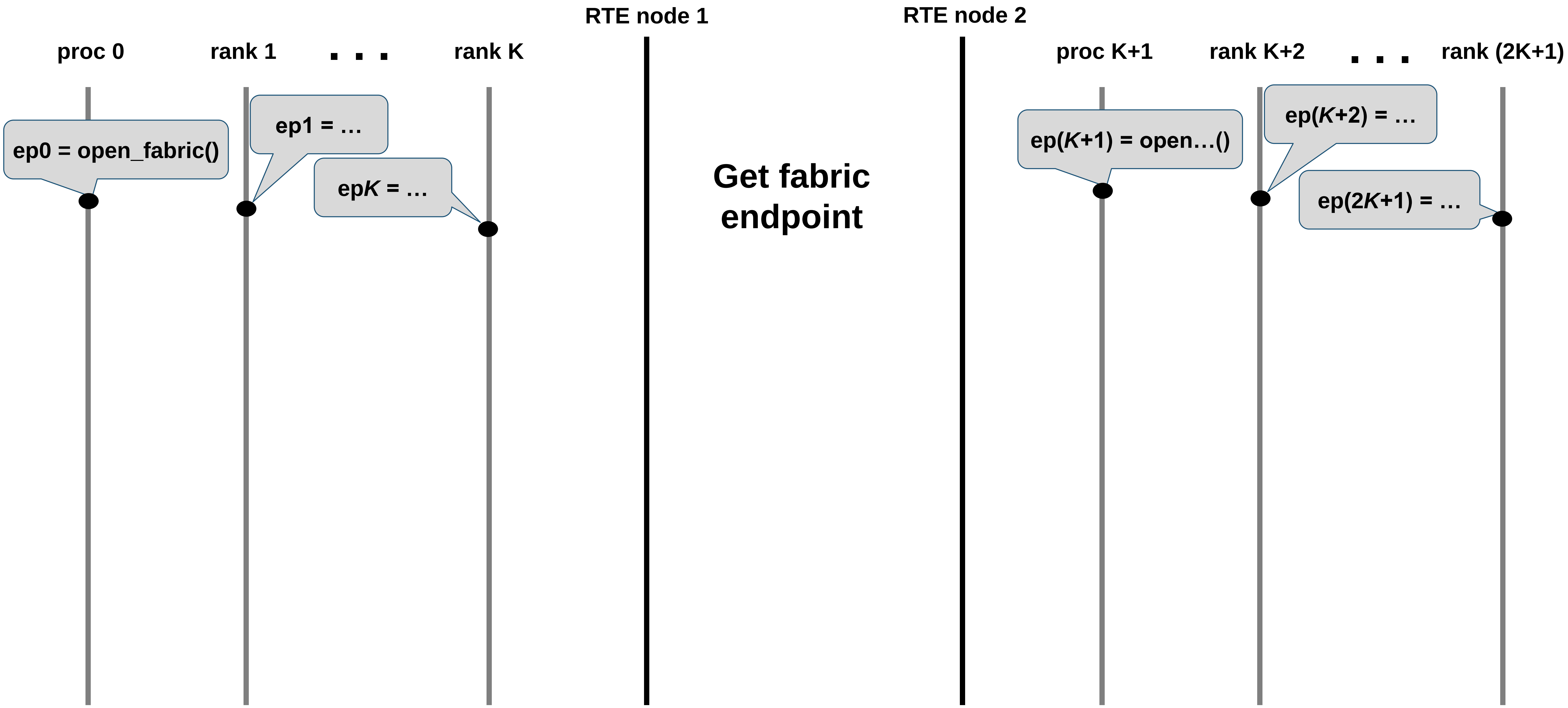
Process Management Interface: RTE – application



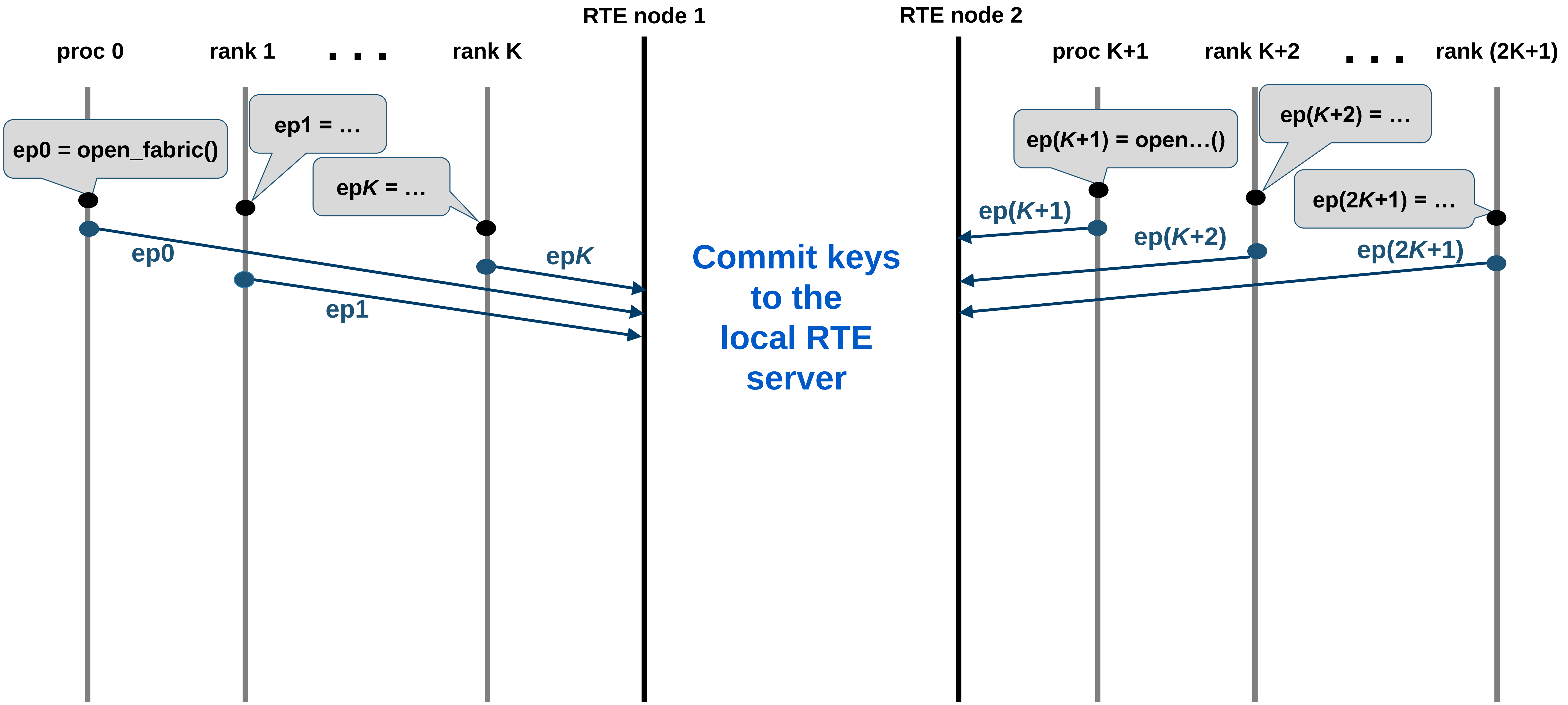
PMIx endpoint exchange modes: full modex



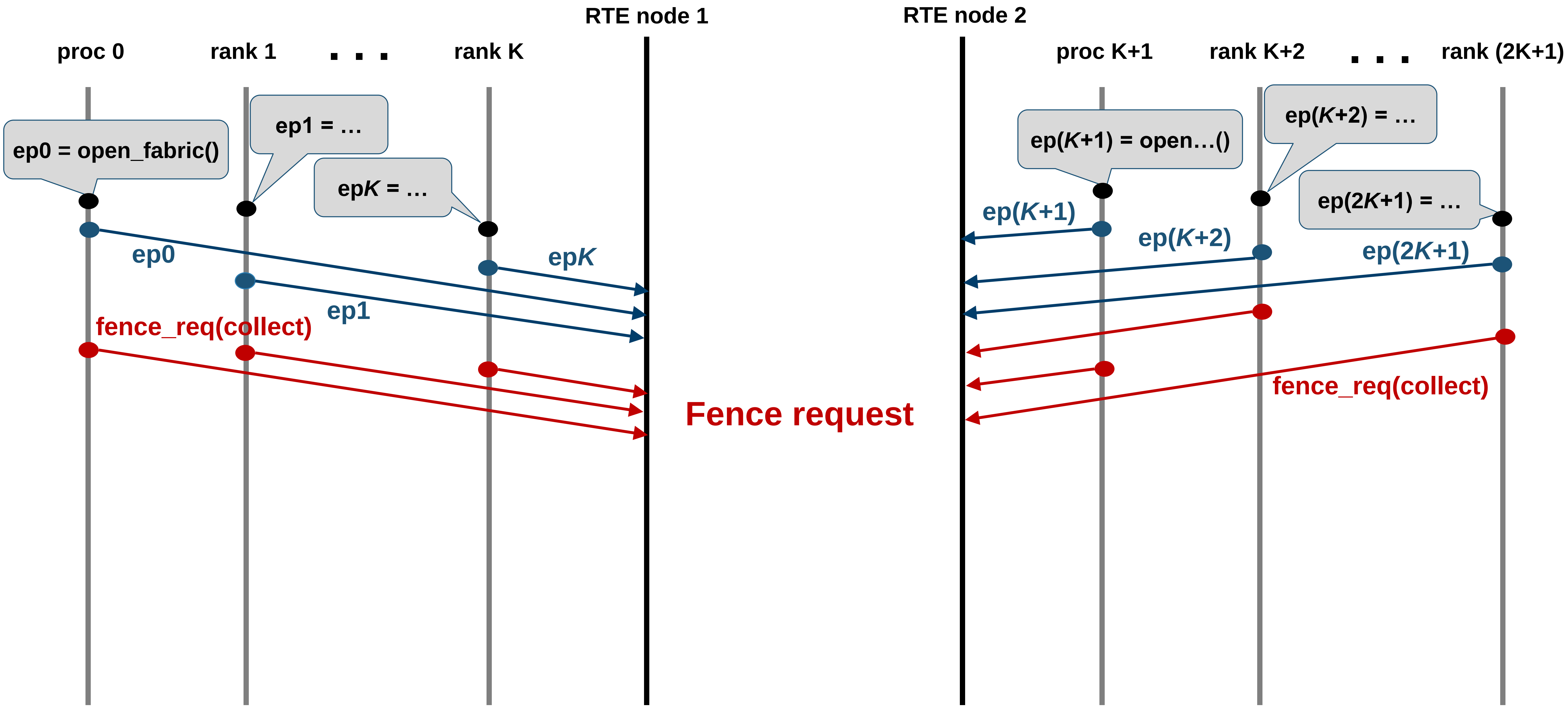
PMIx endpoint exchange modes: full modex (2)



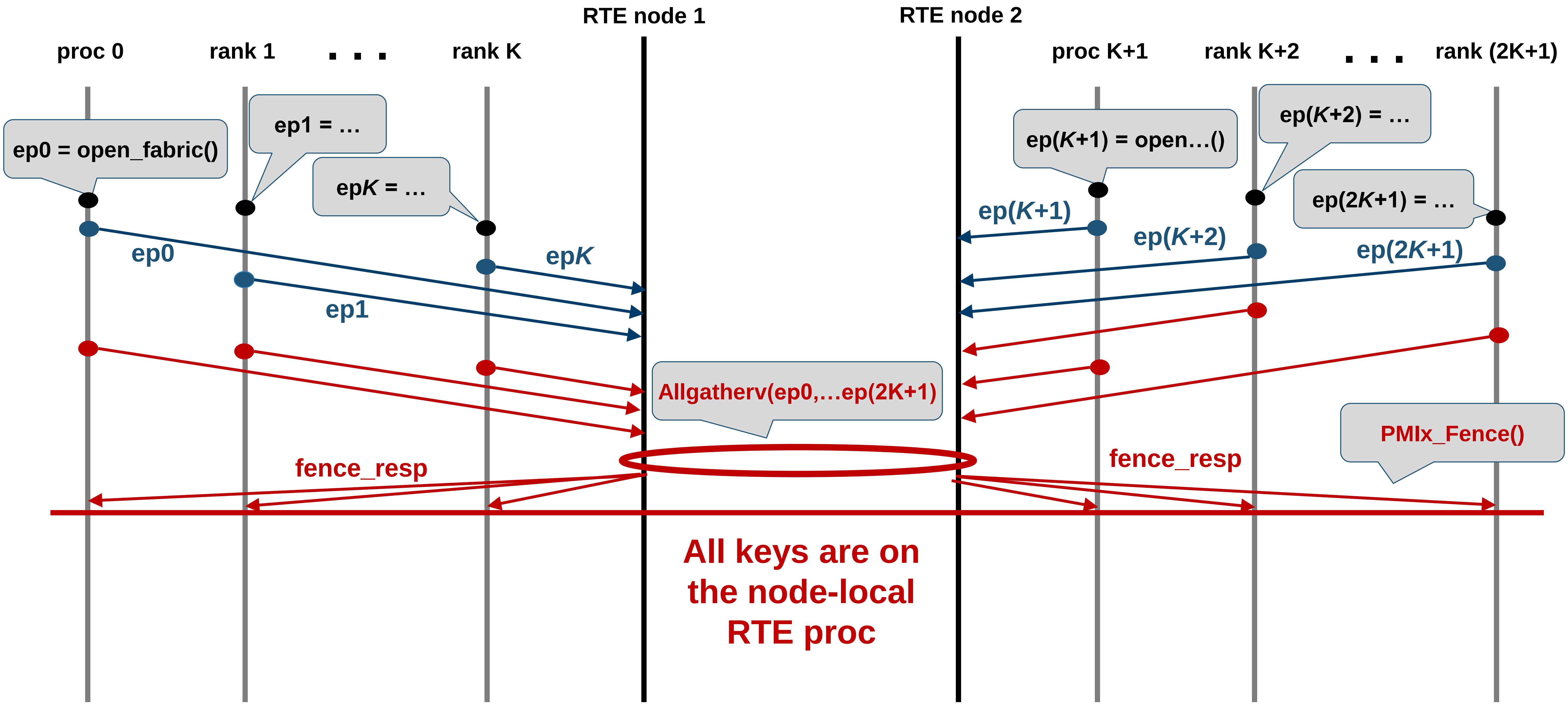
PMIx endpoint exchange modes: full modex (3)



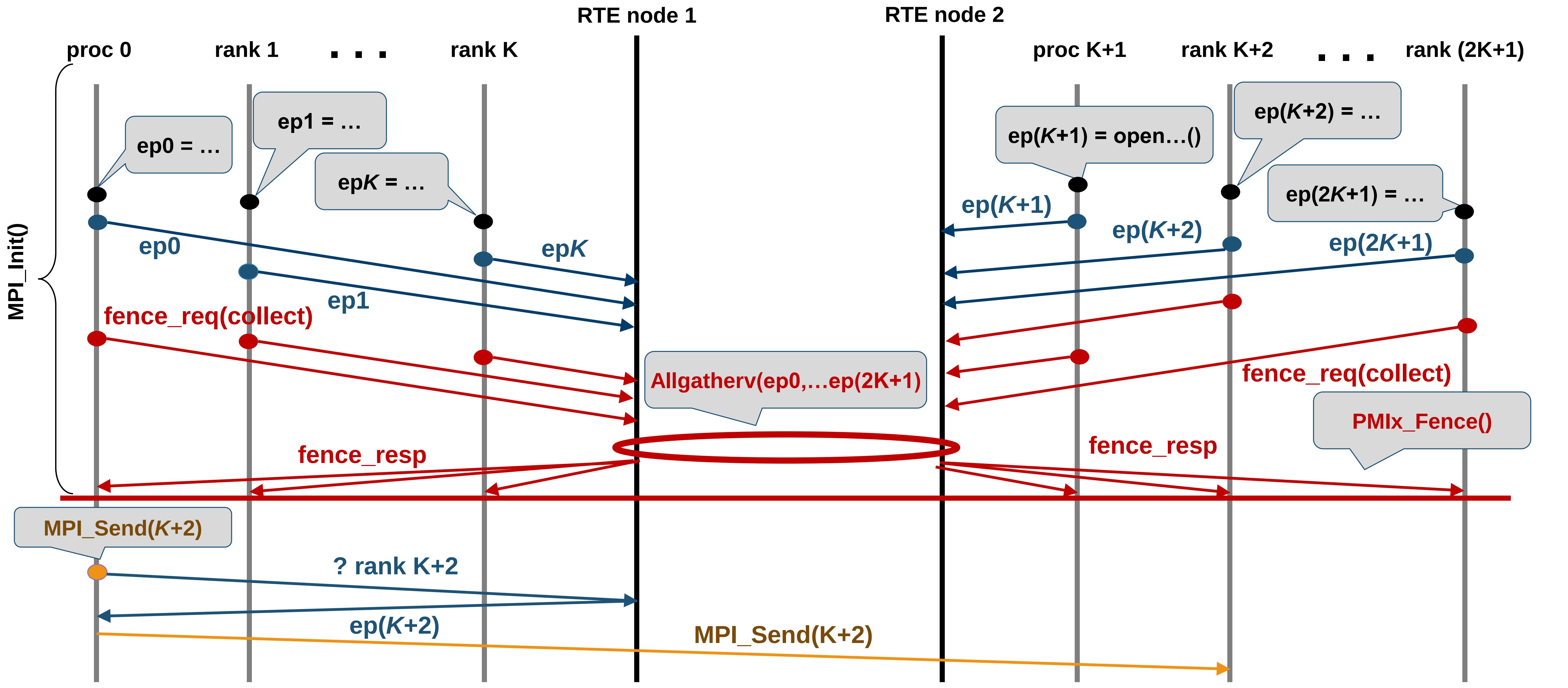
PMIx endpoint exchange modes: full modex (4)



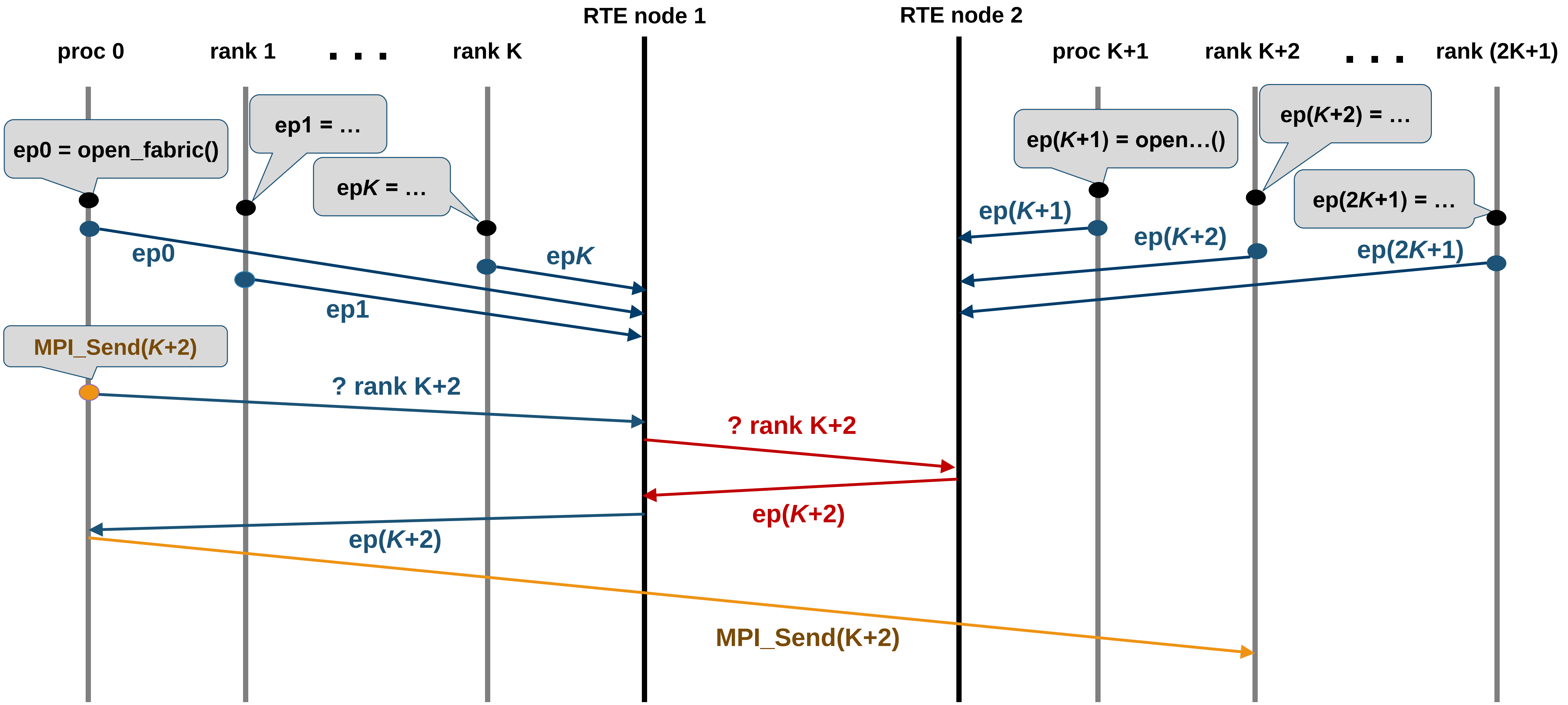
PMIx endpoint exchange modes: full modex (5)



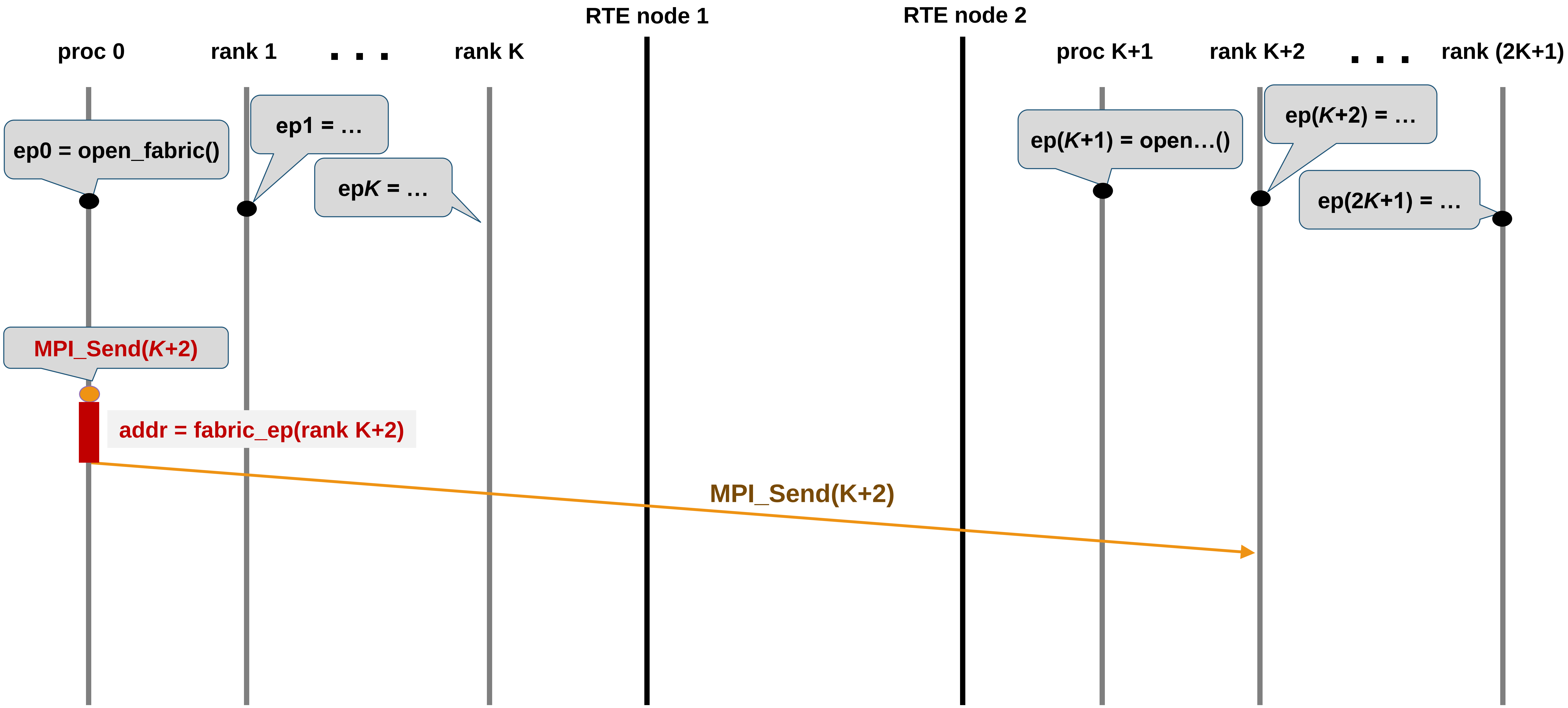
PMIx endpoint exchange modes: full modex (6)



PMIx endpoint exchange modes: direct modex



PMIx endpoint exchange modes: instant-on (future)

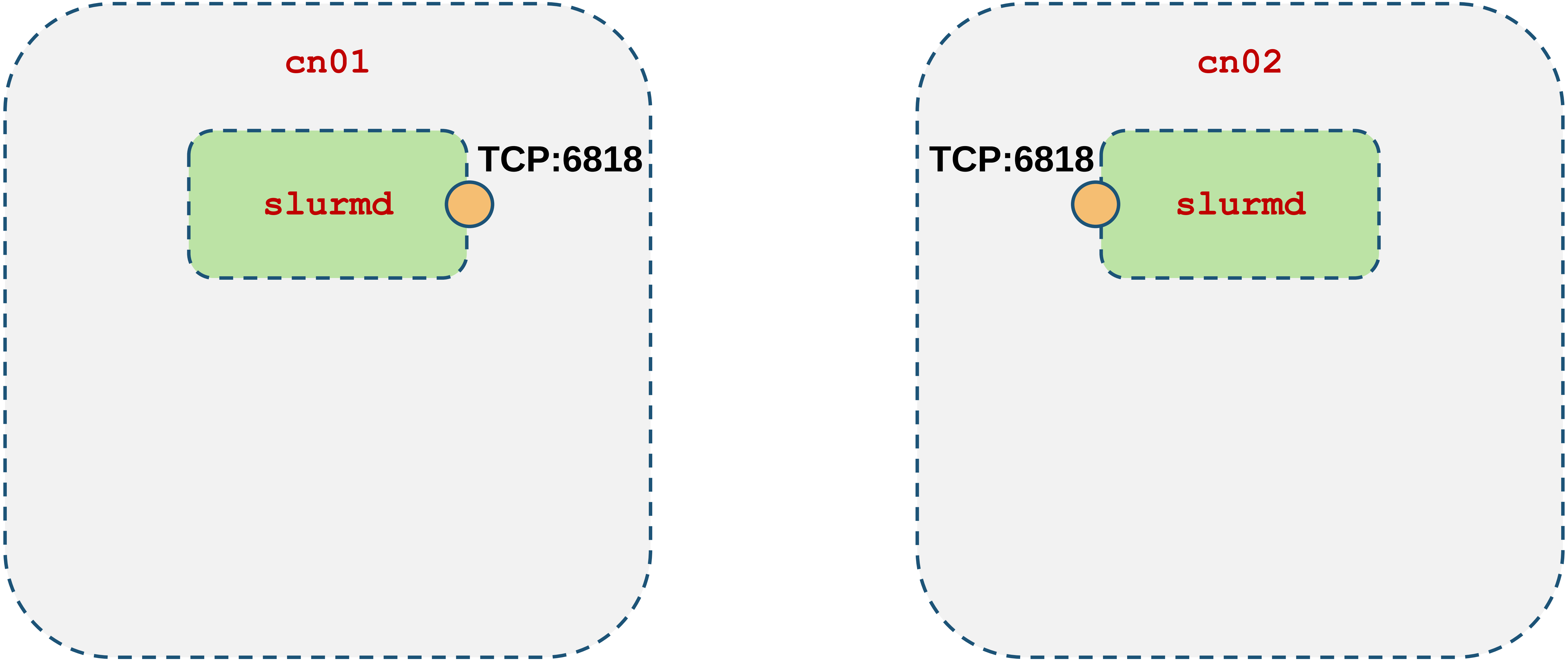


Agenda

- **Problem description**
  - Slurm PMIx plugin status update
  - Motivation of this work
- **What is PMIx?**
  - RunTime Enviroment (RTE)
  - Process Management Interface (PMI)
  - PMIx endpoint exchange modes: full modex, direct modex, instant-on
- **PMIx plugin (Slurm 16.05)**
  - High level overview of a Slurm RPC & analysis.
- **PMIx plugin (Slurm 17.11) – revamp of OOB channel**
  - Direct-connect feature
  - Revamp of PMIx plugin collectives
  - Early wireup feature
  - Performance results for Open MPI

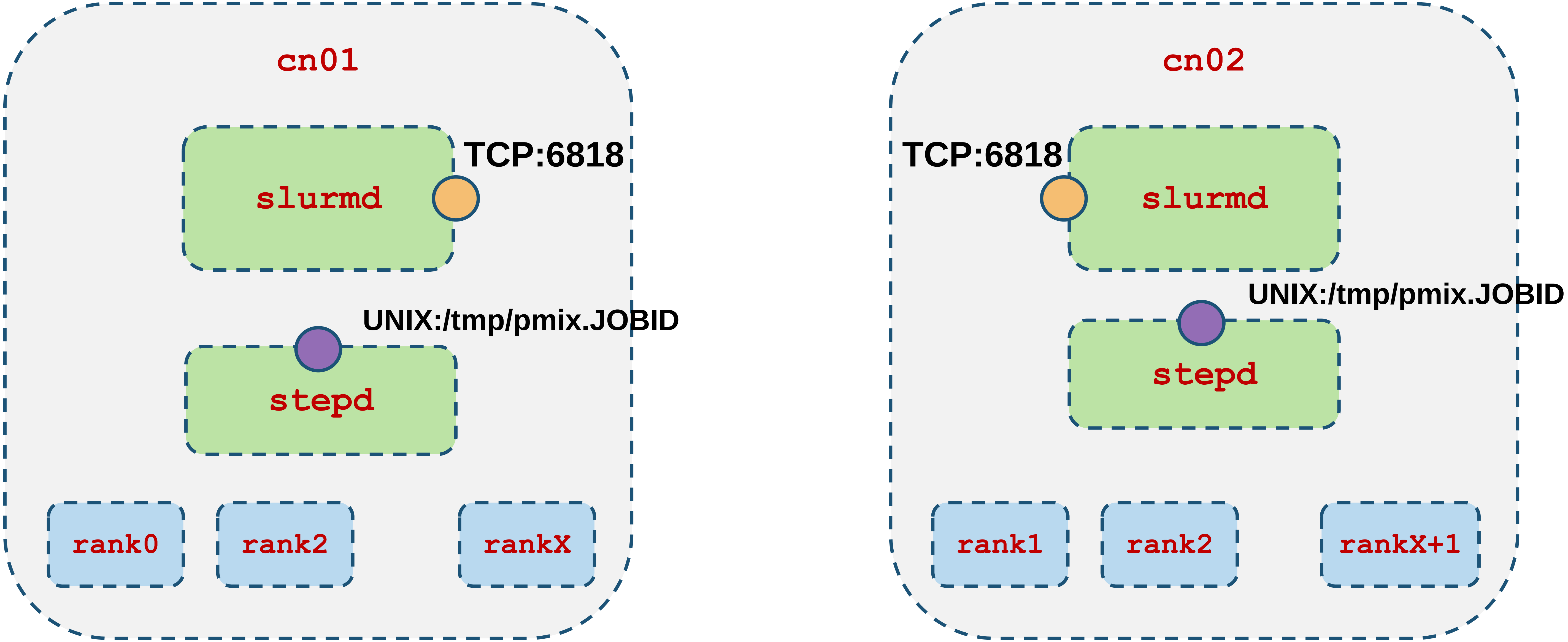
Slurm RPC workflow (1)

Every node has slurmd daemon that controls it. It has a well-known TCP port that allows other components to communicate with it.



Slurm RPC workflow (2)

When a job is launched a SLURM step daemon (stepd) is used to control application processes. stepd also runs the instance of the PMIx server. stepd opens and listens for a UNIX socket.



Slurm RPC workflow (3)

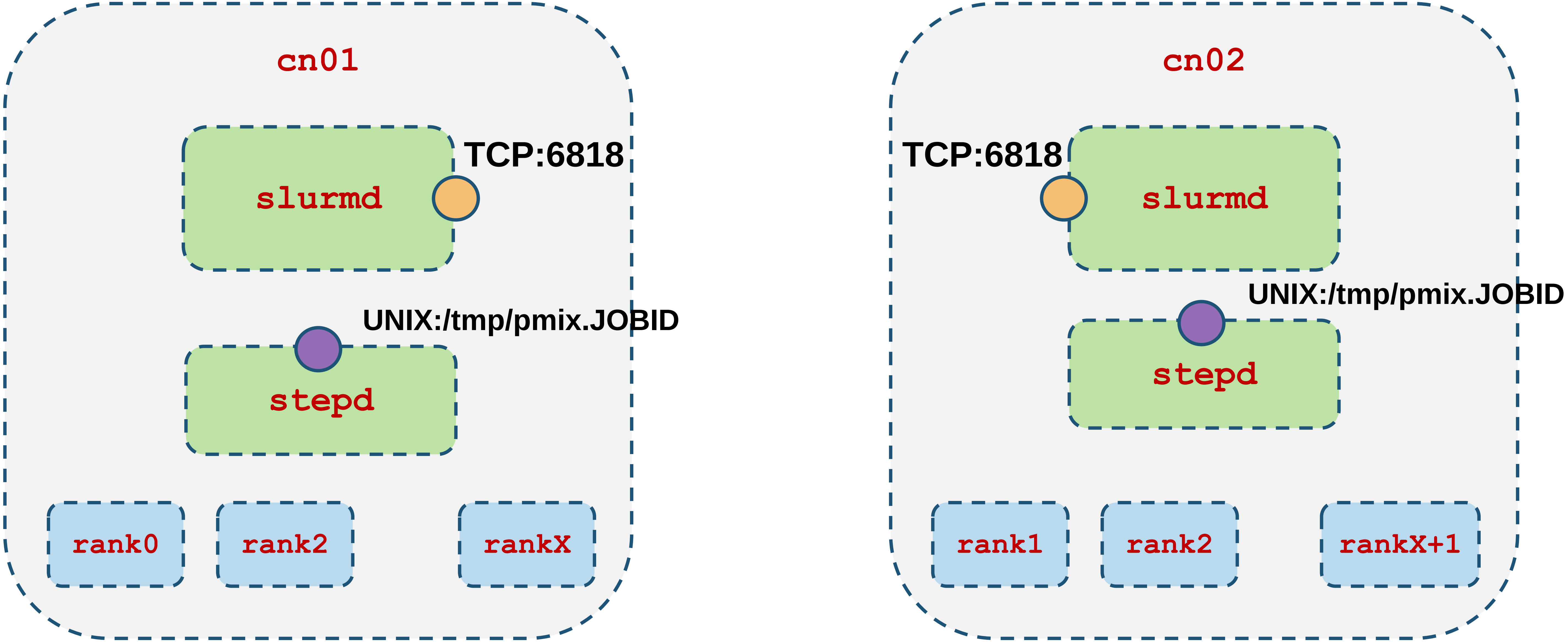
SLURM provides RPC API that allows an easy way to communicate a process on the remote node without connection establishment:

```
slurm_forward_data(nodelist, usock_path, len, data)
```

`nodelist`	SLURM representation of nodenames: **cn[01-10,20-30]**
`usock_path`	path to a UNIX socket file that the process you are trying to reach is listening **/tmp/pmix.JOBID**
`len`	length of a data buffer
`data`	pointer to a data buffer

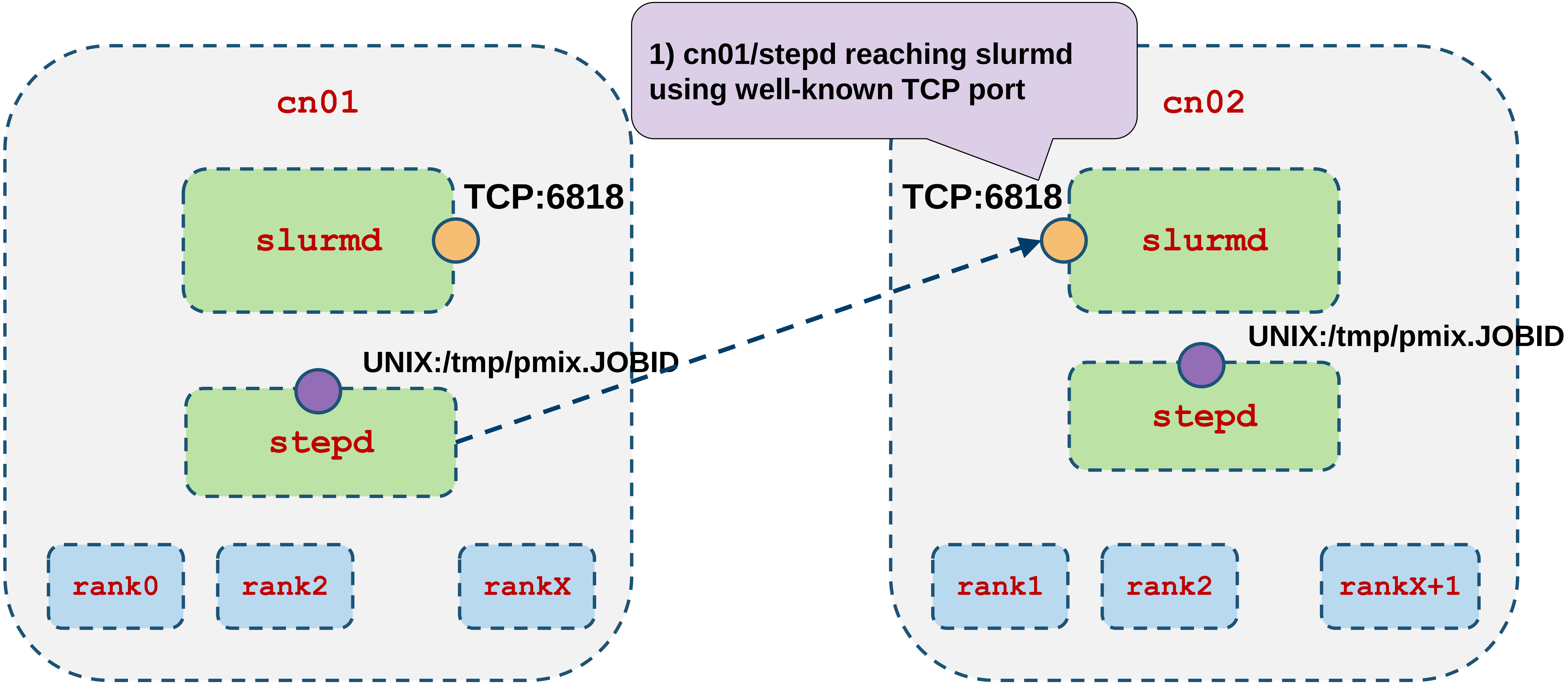
Slurm RPC workflow (4)

`cn01: slurm_forward_data("cn02", "/tmp/pmix.JOBID", len, data)`



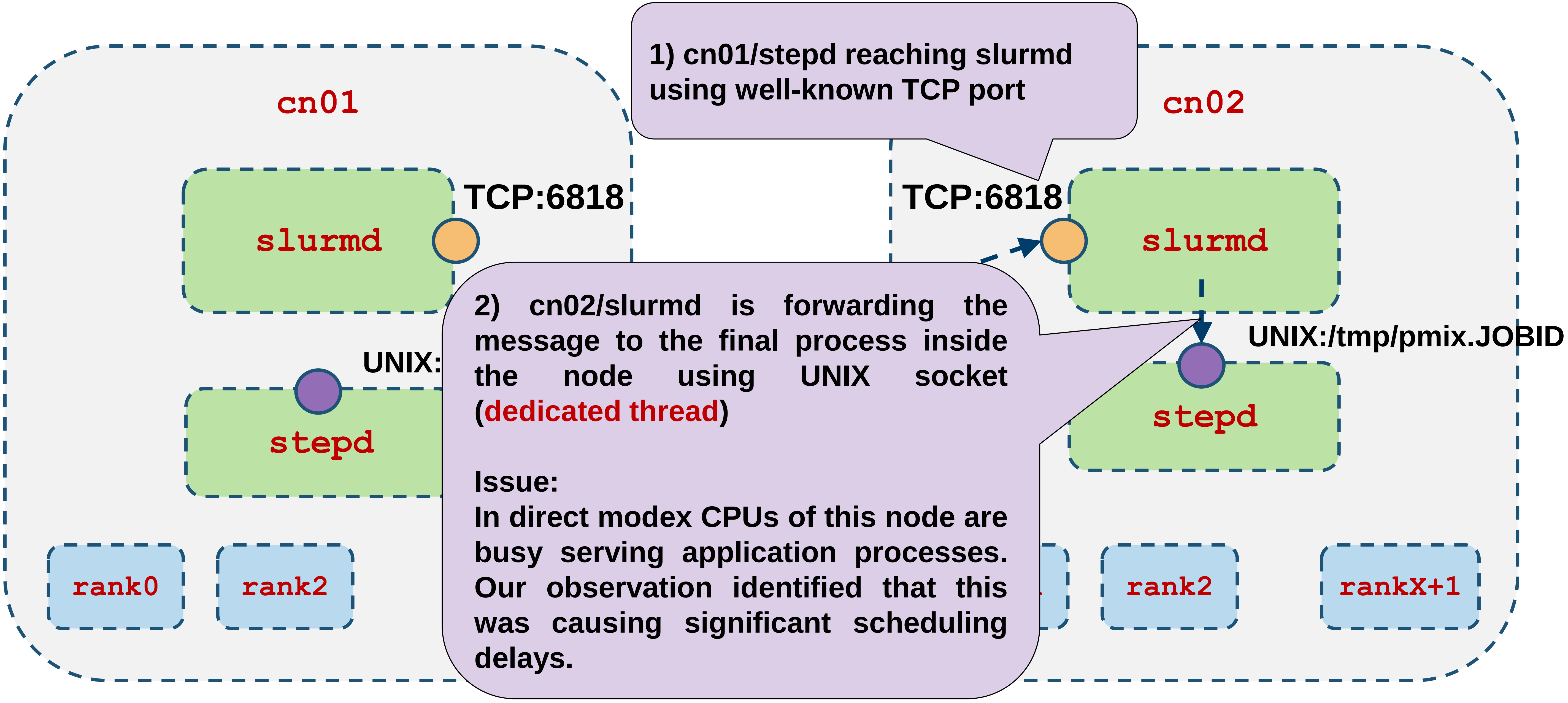
Slurm RPC workflow (5)

`cn01: slurm_forward_data("cn02", "/tmp/pmix.JOBID", len, data)`



Slurm RPC workflow (6)

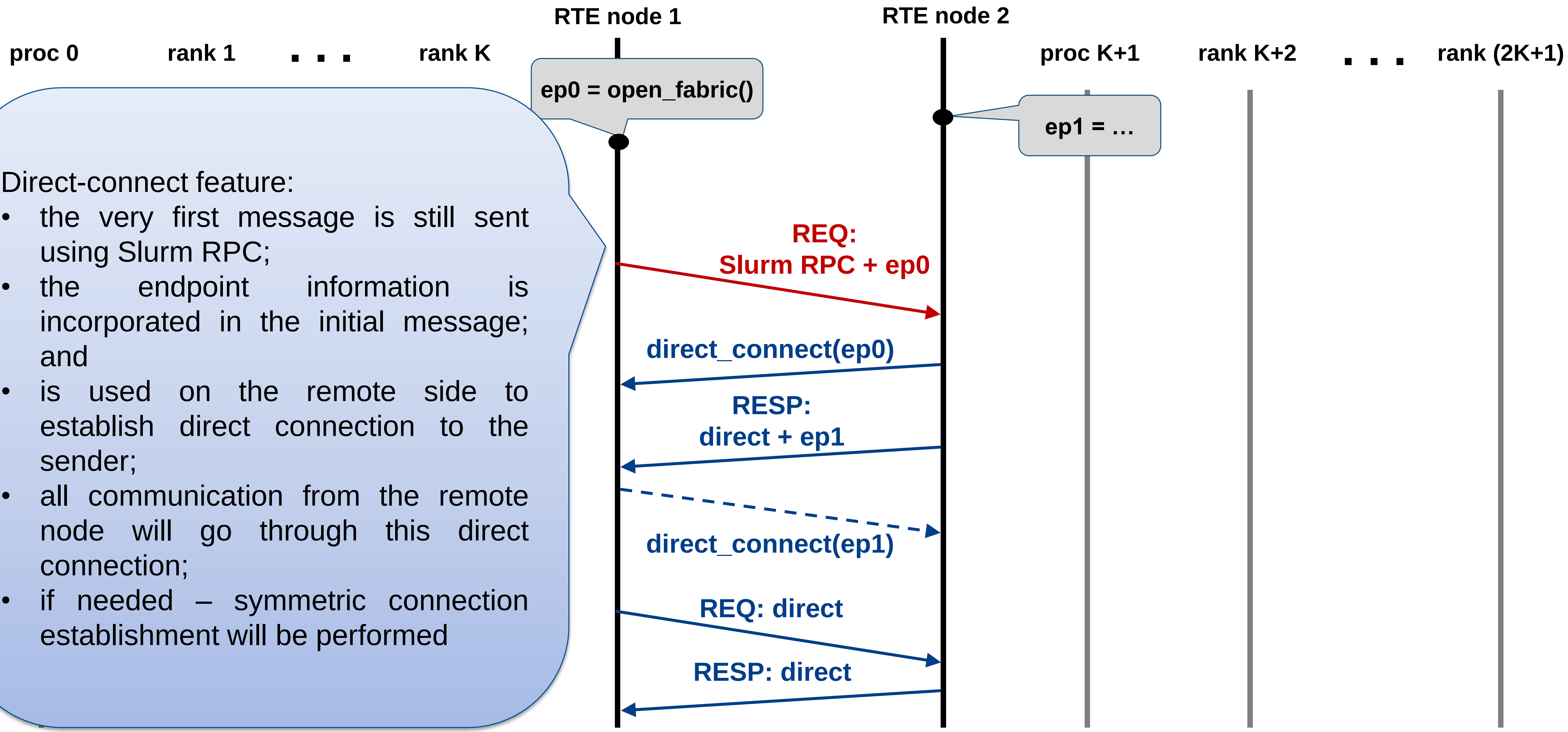
`cn01: slurm_forward_data("cn02", "/tmp/pmix.JOBID", len, data)`



Agenda

- **Problem description**
  - Slurm PMIx plugin status update
  - Motivation of this work
- **What is PMIx?**
  - RunTime Enviroment (RTE)
  - Process Management Interface (PMI)
  - PMIx endpoint exchange modes: full modex, direct modex, instant-on
- **PMIx plugin (Slurm 16.05)**
  - High level overview of a Slurm RPC & analysis.
- **PMIx plugin (Slurm 17.11) – revamp of OOB channel**
  - Direct-connect feature
  - Revamp of PMIx plugin collectives
  - Early wireup feature
  - Performance results for Open MPI

Direct-connect feature



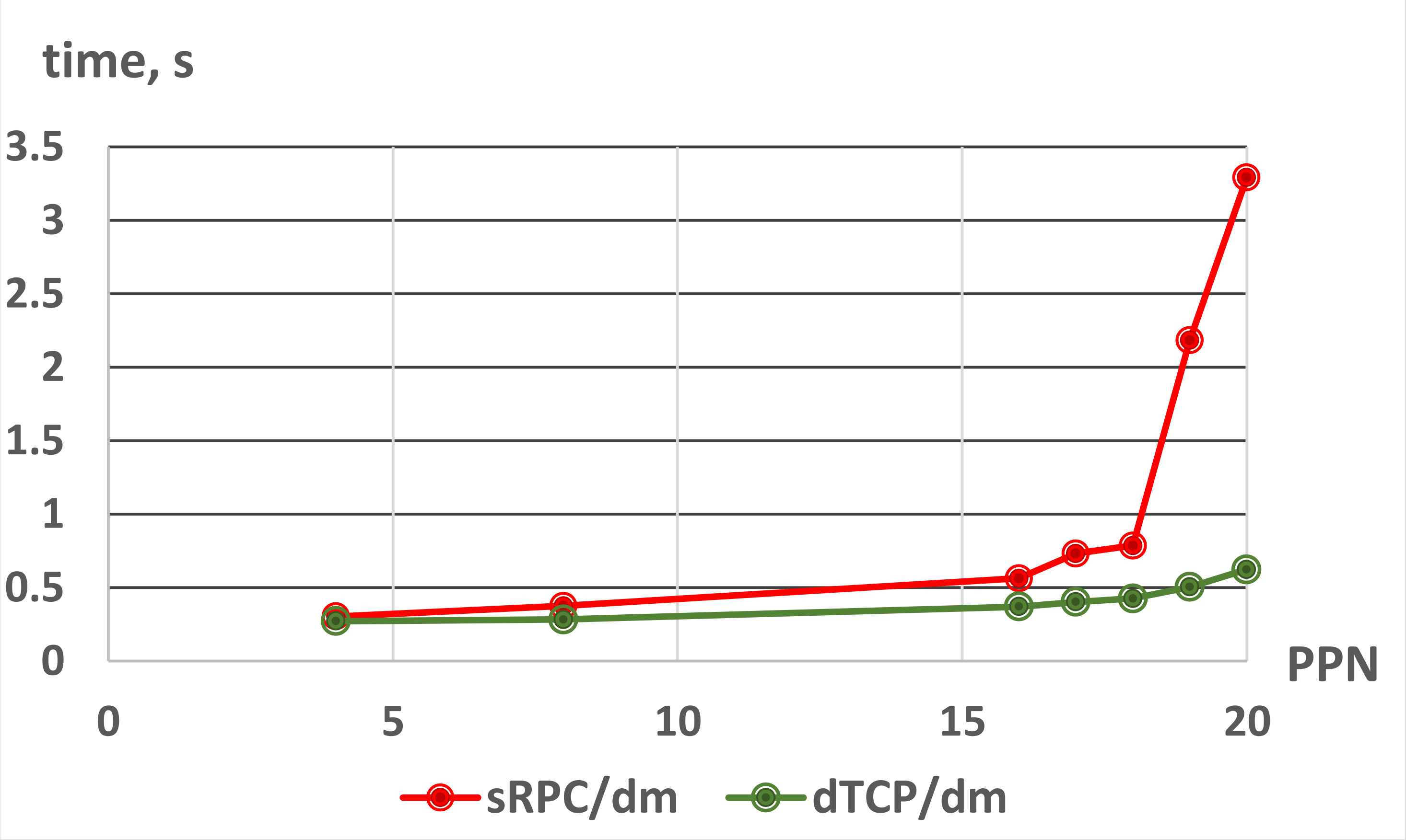
Direct-connect feature:

- the very first message is still sent using Slurm RPC;
- the endpoint information is incorporated in the initial message; and
- is used on the remote side to establish direct connection to the sender;
- all communication from the remote node will go through this direct connection;
- if needed – symmetric connection establishment will be performed

Direct-connect feature: TCP-based

- The first version of direct-connect was TCP-based
- Slurm RPC is still supported, but needs to be enabled using **`SLURM_PMIX_DIRECT_CONN`** environment variable.

The performance of the OpenSHMEM jobstart was significantly improved. Below is the time to perform shmem_init() on 32 nodes with various Process Per Node (PPN) count. sRPC stands for Slurm RPC, dTCP – TCP-based direct-connect.



- Related environment variables:

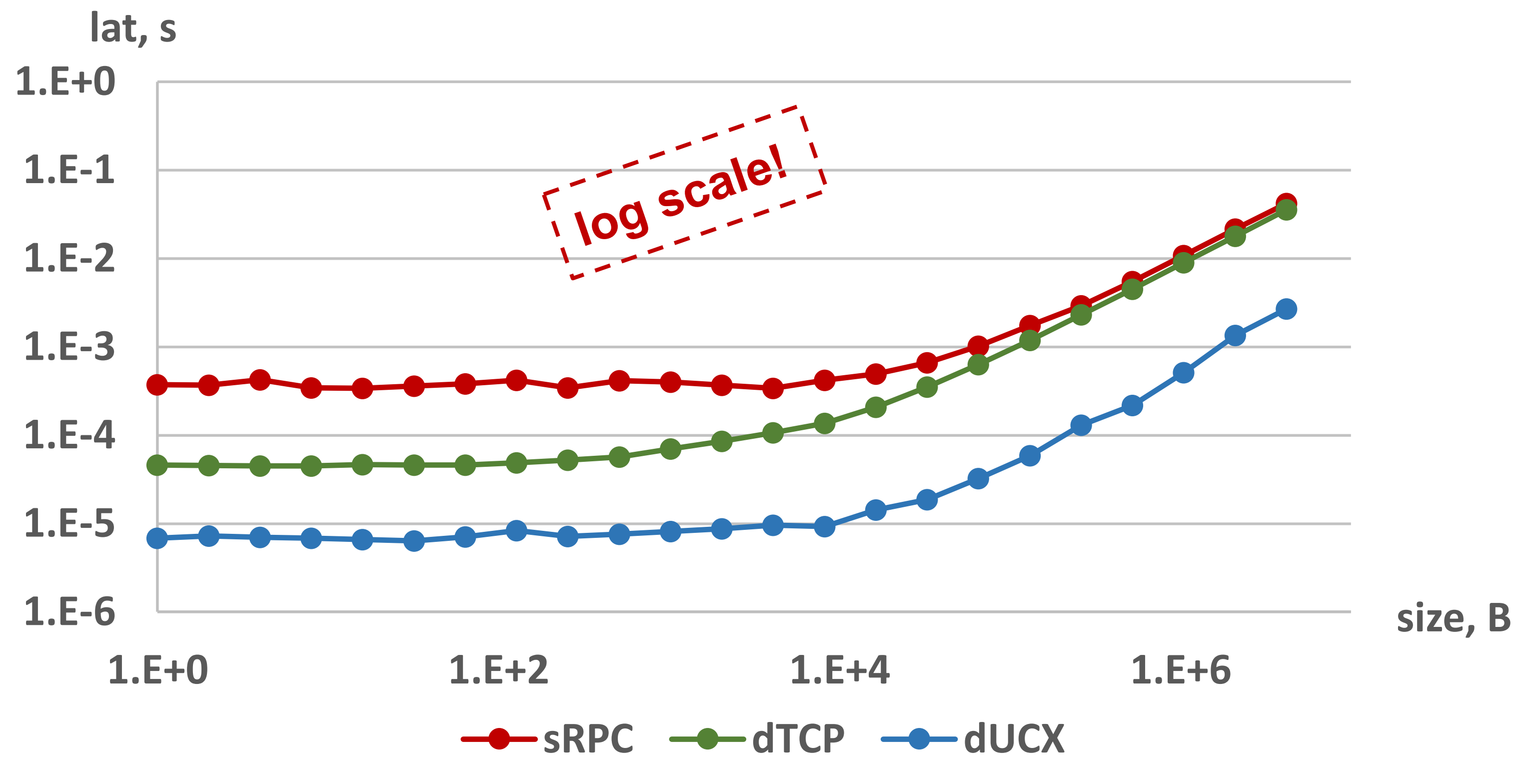
`SLURM_PMIX_DIRECT_CONN = { true | false }`

Enables direct connect, (true by default)

Direct-connect feature: UCX-based

- Existing direct-connect infrastructure allowed to use HPC fabric for communication.
- Support for UCX porint-to-point communication library (www.openucx.com) was implemented.
- Slurm 17.11 should be configured with “--with-ucx=<ucx-path>” to enable UCX support.

Below is the latency measured for the point-to-point exchange* for each of the communication options available in Slurm 17.11: (**a**) Slurm RPC (sRPC); (**b**) TCP-based direct-connect (dTCP); (**c**) UCX-based direct-connect (dUCX).



- Related environment variables:

SLURM_PMIX_DIRECT_CONN = {true | false}

Enables direct connect, (true by default)

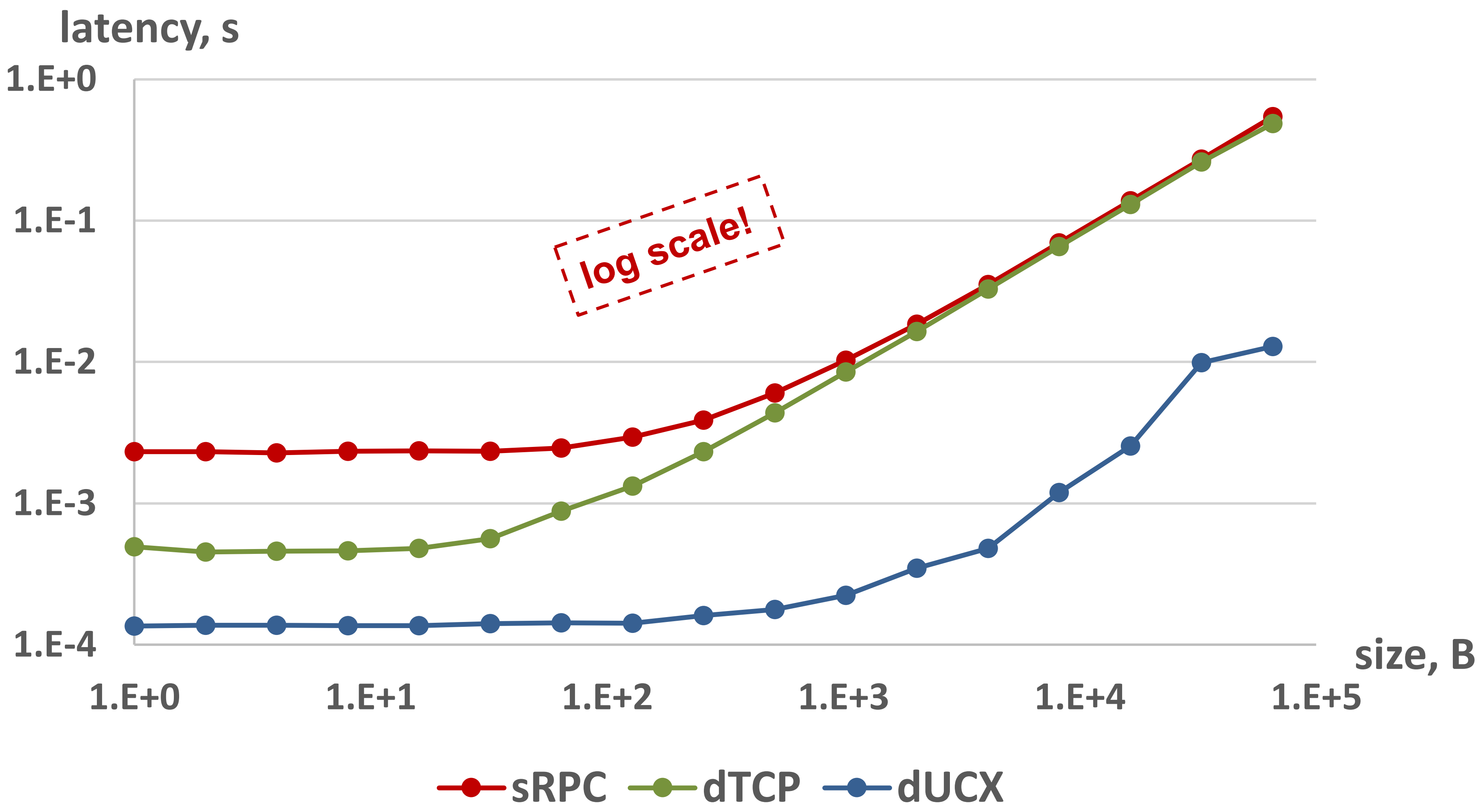
SLURM_PMIX_DIRECT_CONN_UCX = {true | false}

Enables direct connect, (true by default)

* See the backup slide #1 for the details about point-to-point benchmark

Revamp of PMIx plugin collectives

- PMIx plugin collectives infrastructure was also redesigned to leverage direct-connect feature.
- The results of a collective micro-benchmark (see backup slide #2) for 32-node cluster (one stepd per node) are provided below:



* See the backup slide #2 for the details about collectives benchmark

Early wireup feature

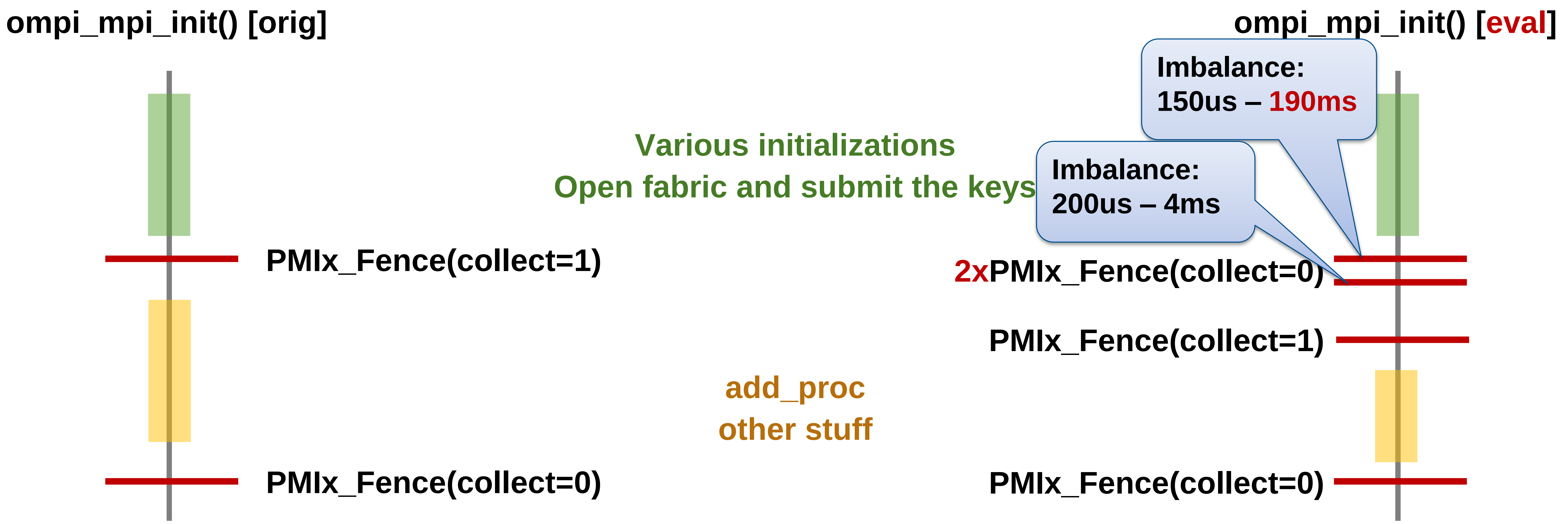
- Implementation of the direct-connect assumes that Slurm RPC is still used for the address exchange.
- This address exchange is initiated at the first communication.
- This is an issue for PMIx full modex mode, because the first communication is usually the heaviest (Allgatherv).
- To deal with that an early-wireup feature was introduced.
  - The main idea is that step daemons start wiring up right after they were launched without waiting for the first communication.
  - Open MPI as an example usually does some local initialization that provides a reasonable room to perform the wireup in the background.

Related environment variables:

`SLURM_PMIX_DIRECT_CONN_EARLY = {true | false}`

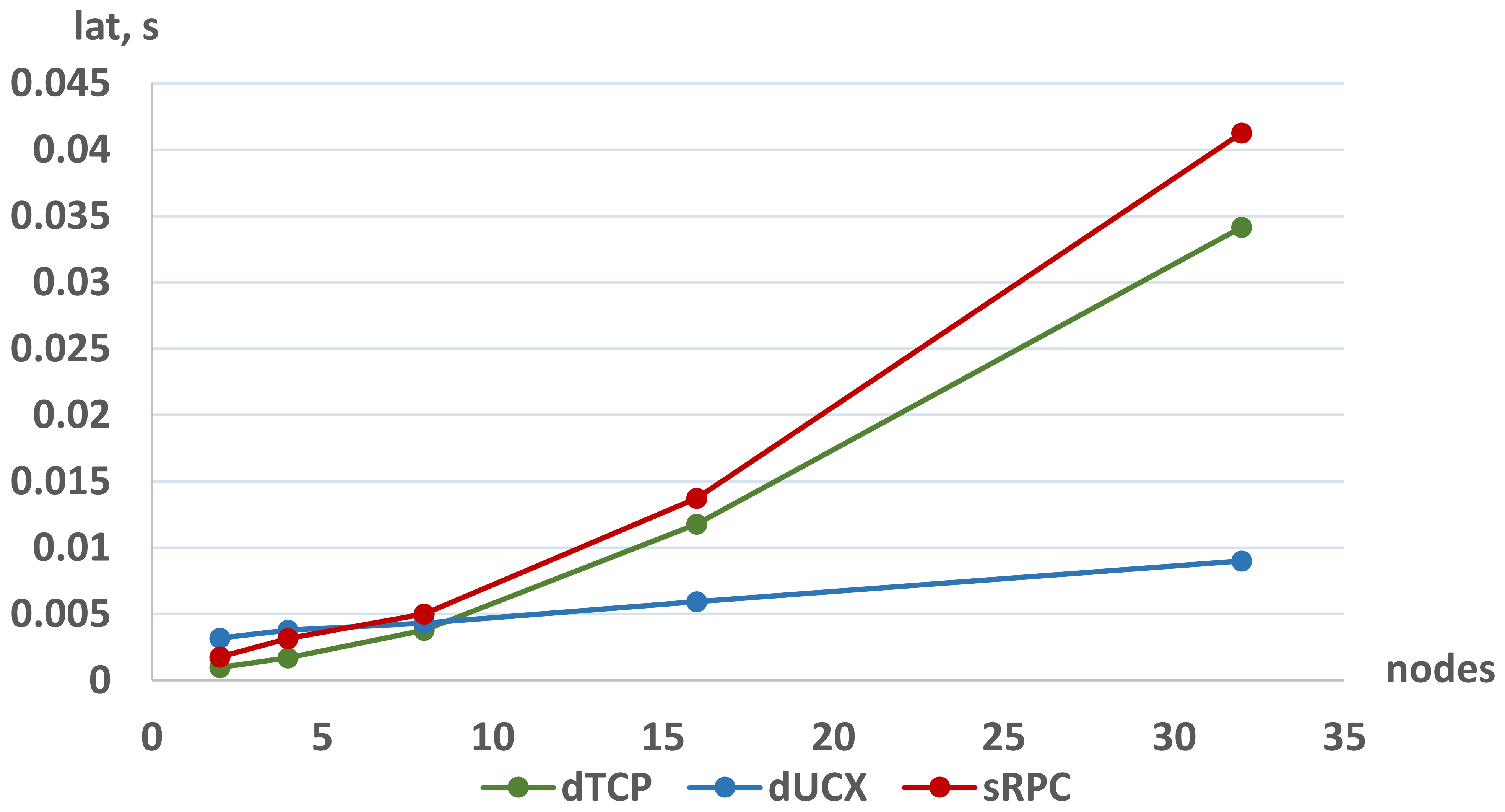
Performance results for Open MPI modex

- At the small scale the latency of PMIx_Fence() is affected by the processes imbalance.
- To get the clear numbers we modified Open MPI ompi_mpi_init function by adding 2 additional PMIx_Barrier()s as shown on the diagram below:



Performance results for Open MPI modex (2)

Below is the dependency of an average of a maximum time spent in PMIx_Fence(collect=1) relative to the number of nodes is presented:



Measurements configuration:

- 32 nodes, 20 cores per node
- PPN = 20
- PMIx v1.2
- OMPI v2.x
- Slurm 17.11 (pre-release)

Future work

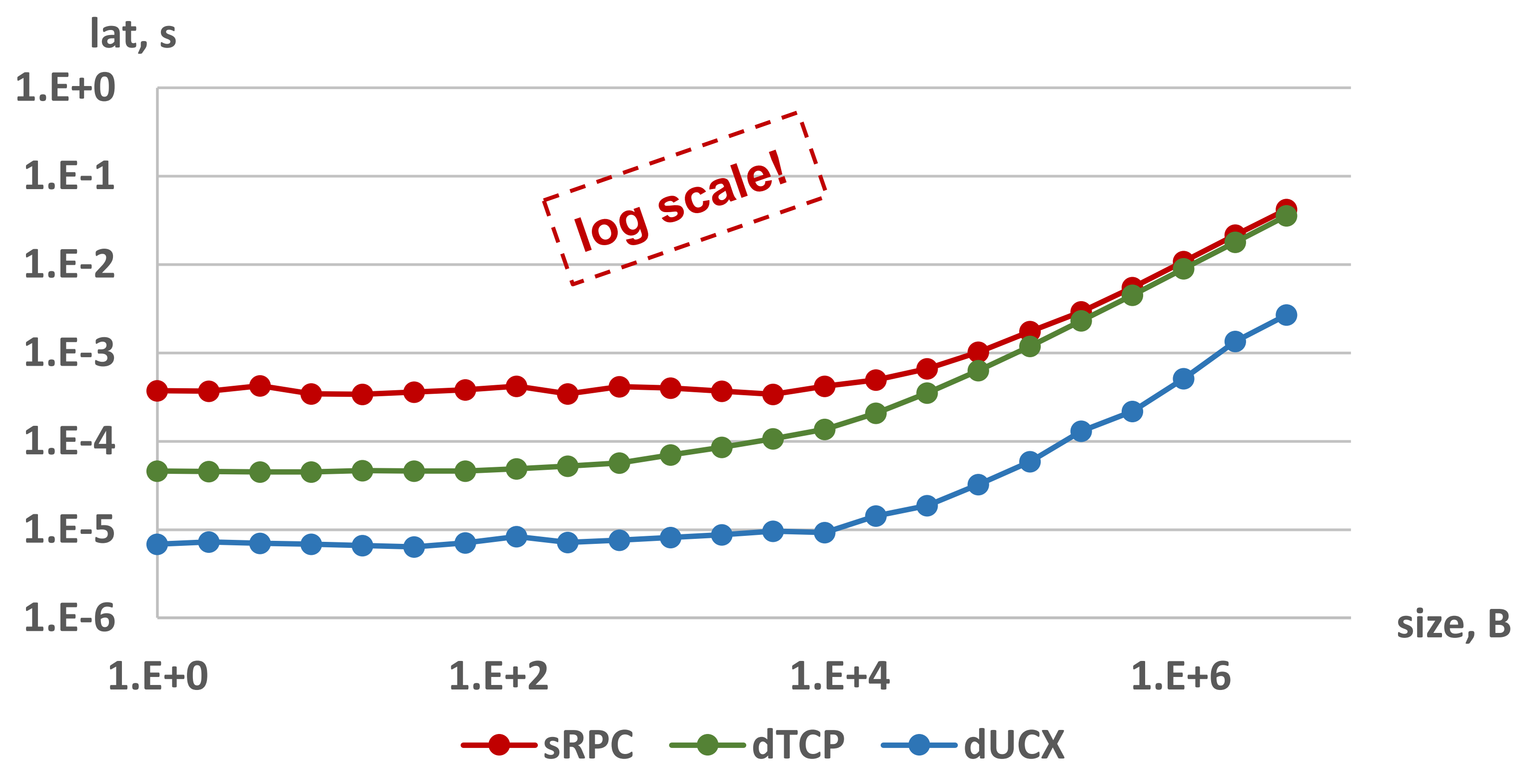
- Need wider testing of new features
  - Let us know if you have any issues: artemp [at] mellanox.com
- Scaling tests and performance analysis
  - Need to evaluate efficiency of early wireup feature
- Analyze possible impacts on other jobstart stages:
  - Propagation of the Slurm launch message (deviation ~2ms).
  - Initialization of the PMIx and UCX libraries (local overhead)
  - Impact of UCX used for resource management on application processes
  - Impact of local PMIx overhead
- Use this feature as an intermediate stage for instant-on
  - Pre-calculate job’s stepd endpoint information and use UCX to exchange endpoint info for application processes.

Thank You

Mellanox TECHNOLOGIES
Connect. Accelerate. Outperform.™

Integrated point-to-point micro-benchmark (backup#1)

- To estimate the point-to-point latency of available transports the point-to-point micro-benchmark was introduced in Slurm PMIx plugin.
- To activate it, Slurm must be configured with "--enable-debug" option.

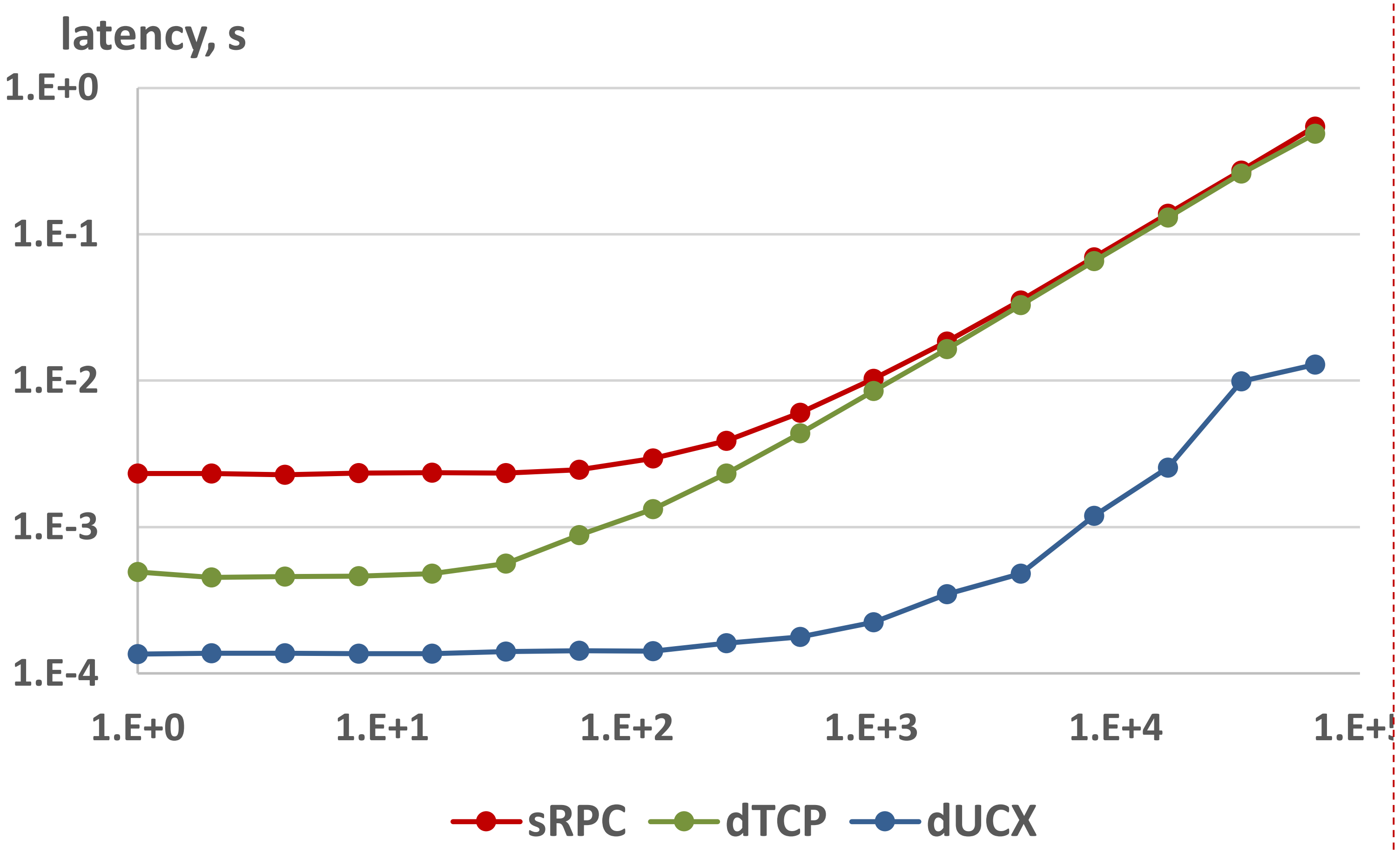


Related environment variables:

```
SLURM_PMIX_WANT_PP=1
        Turn point-to-point benchmark on
SLURM_PMIX_PP_LOW_PWR2=0
SLURM_PMIX_PP_UP_PWR2=22
        Message size range (powers of 2)
        from 1 to 4194304 in this example
SLURM_PMIX_PP_ITER_SMALL=100
        Number of iterations for small messages
SLURM_PMIX_PP_ITER_LARGE=20
        Number of iterations for large messages
SLURM_PMIX_PP_LARGE_PWR2=10
        Switch to the large message starting
        from 2^val
```

Collective micro-benchmark (backup #2)

- PMIx plugin collectives infrastructure was also redesigned to leverage direct-connect feature.
- The results of a collective micro-benchmark for 32-node cluster (one stepd per node) are provided below:



Related environment variables:

```
SLURM_PMIX_WANT_COLL_PERF=1
        Turn collective benchmark on
SLURM_PMIX_COLL_PERF_LOW_PWR2=0
SLURM_PMIX_COLL_PERF_UP_PWR2=22
        Message size range (powers of 2)
        from 1 to 65536 in this example
SLURM_PMIX_COLL_PERF_ITER_SMALL=100
        Number of iterations for small messages
SLURM_PMIX_COLL_PERF_ITER_LARGE=20
        Number of iterations for large messages
SLURM_PMIX_COLL_PERF_LARGE_PWR2=10
        Switch to the large message starting
        from 2^val
```