DMTCP: Fixing the Single Point of Failure of the ROS Master

Twinkle Jain
jain.t@husky.neu.edu

Gene Cooperman
gene@ccs.neu.edu

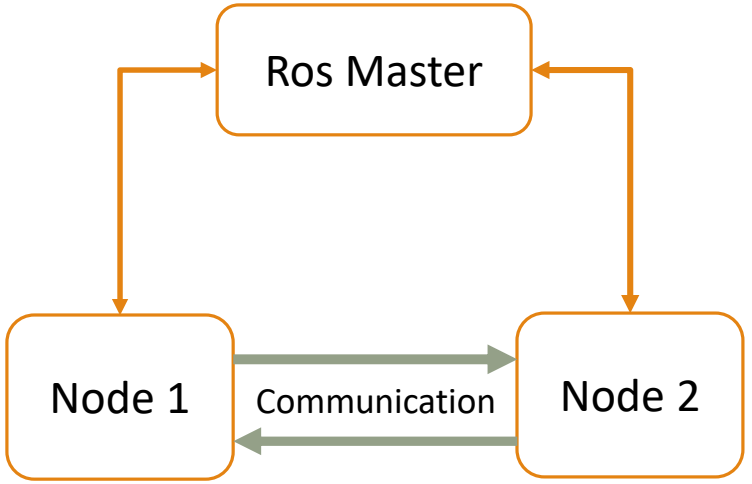
College of Computer and Information Science
Northeastern University, Boston, USA

Roadmap

- Core Problem: ROS master failure
- Possible Solutions
- Effective solution: Transparent checkpointing (DMTCP)
- DMTCP's workflow
- Barrier: DMTCP didn't support Pseudoterminals
- Pseudoterminal plugin
- Extending the DMTCP
- Live demo
- DMTCP's growing number of use cases
- DMTCP, ROS2, and the future!
- Summary

Core Problem: ROS master failure

- ROS master contributes in:
  - Provides name and parameter services to the rest of the nodes
  - Control over system, graphs, nodes
  - Helps a node to discover other nodes
- Why is the ROS master's failure a problem?
  - No way to recover the system after crash
  - Nodes need to reload
  - System reinitialization
  - Whole system will compromise
  - The amount of loss can be huge in a robotics application



- See talk by our collaborator, Matthieu Amy, at ROSCon-2016

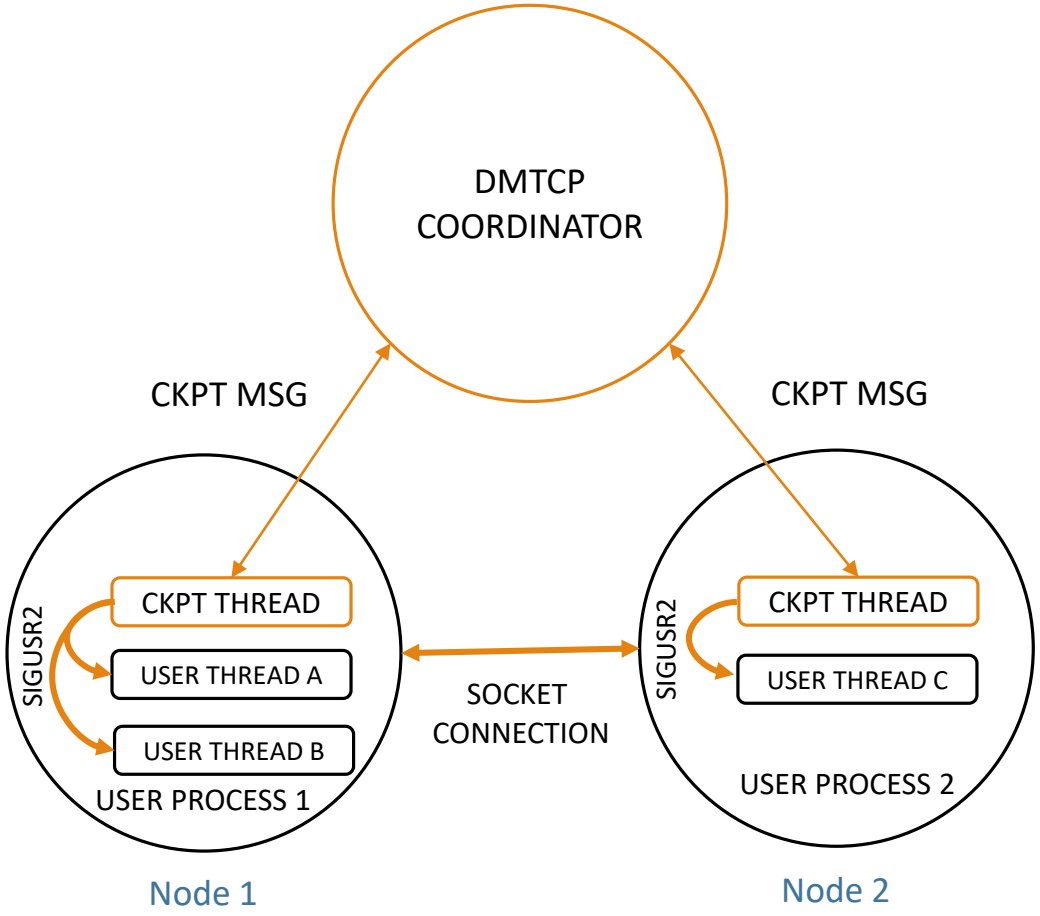
Possible Solutions

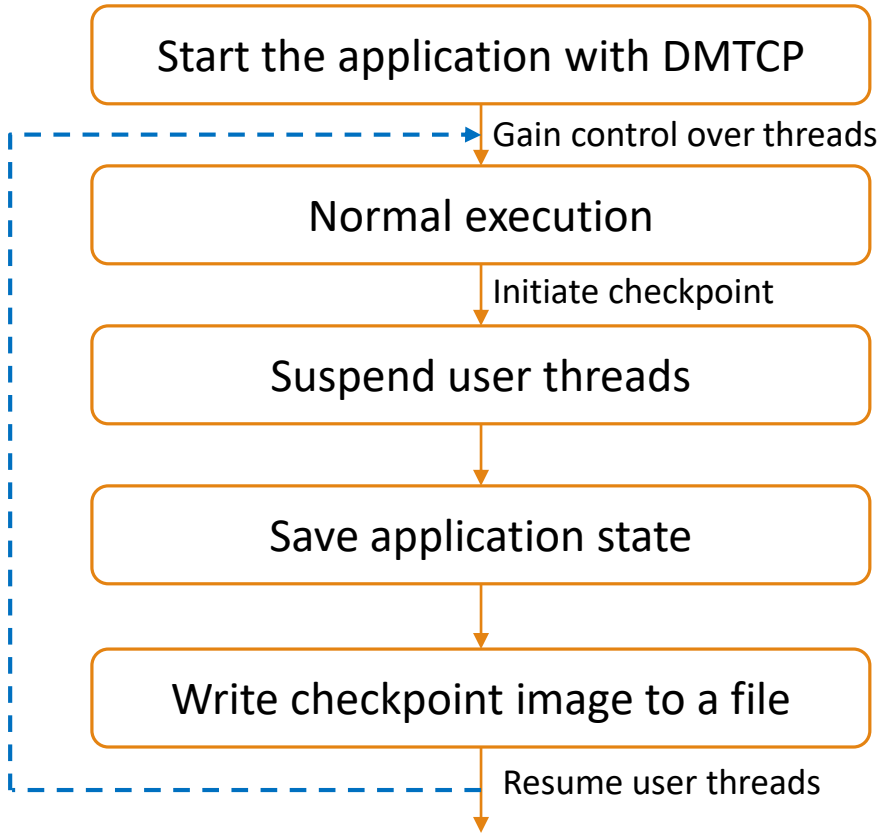
- Switch to a more reliable hardware
  - Expensive solution
  - Chances of failure decreases but crash can happen
- Reduce the dependency on the ROS master
  - Currently unavailable in ROS1
  - Can be applied in the future versions of ROS
- **Checkpoint and Restart (presented here)**
  - Save the state periodically
  - Resume from the last checkpointed state

Effective solution: Transparent Checkpointing (DMTCP)

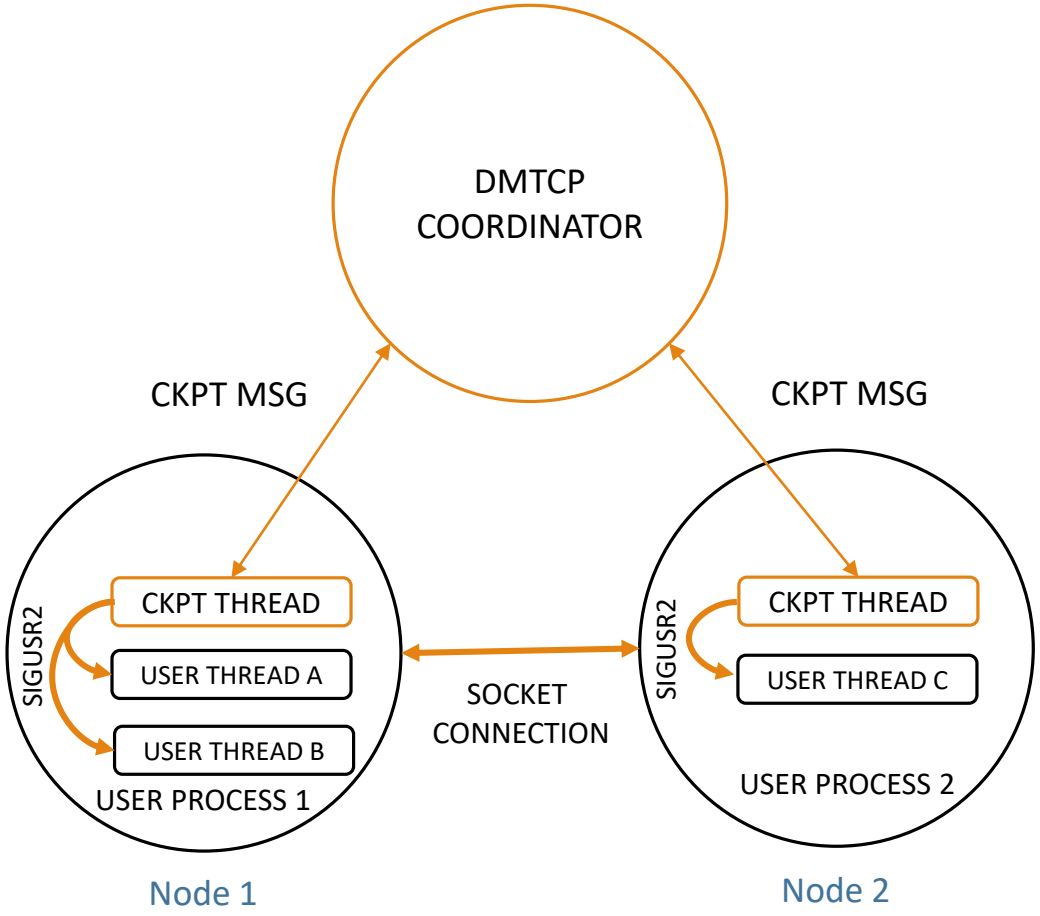
- DMTCP: Distributed Multi-Threaded CheckPointing (http://dmtcp.sourceforge.net/)
  - Active project for 13 years
  - Over 10,000 downloads of source, unkown # downloads of binary package
- **Open source** package for transparent checkpointing
- **Checkpointing** – saving restorable application state like a snapshot
- **Transparent** – application remains unmodified
- **User space** – kernel remains unmodified
- **Easily extensible** – Plugin-based architecture
- Both manual and periodic checkpointing options are available
- NEWS: Being extended to support NVIDIA GPUs

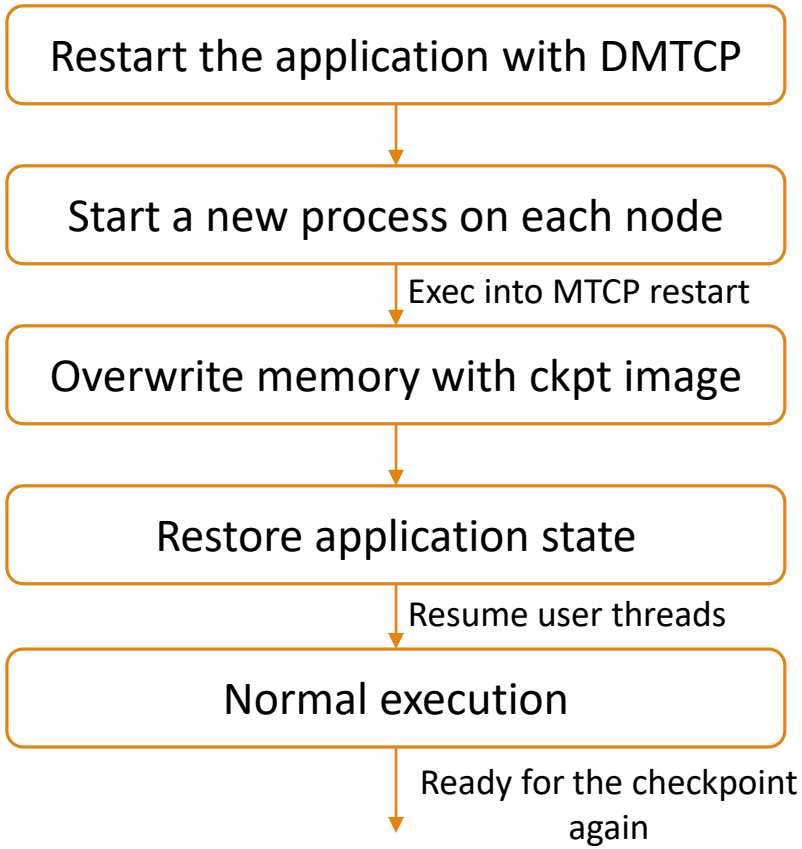
DMTCP's workflow: Checkpointing





DMTCP's workflow: Restart





Barrier: DMTCP didn't support Pseudoterminals!

- When DMTCP tried to checkpoint a ROS application, it didn't work!
- Pseudoterminals
  - Pair of two virtual devices connected with a bidirectional IPC channel
  - Emulates a device terminal for terminal oriented programs (e.g., vi editor)
- What is the problem?
  - A typical ROS application uses ptys
  - DMTCP doesn't handle pty's checkpointing correctly
  - The state of a pty-based application was not same at the time of checkpoint, resume after checkpoint and at restart

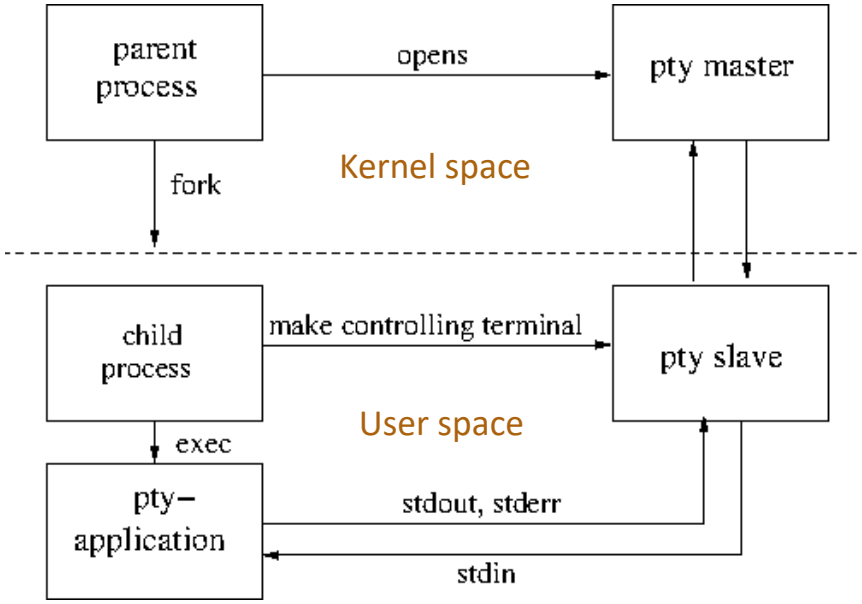
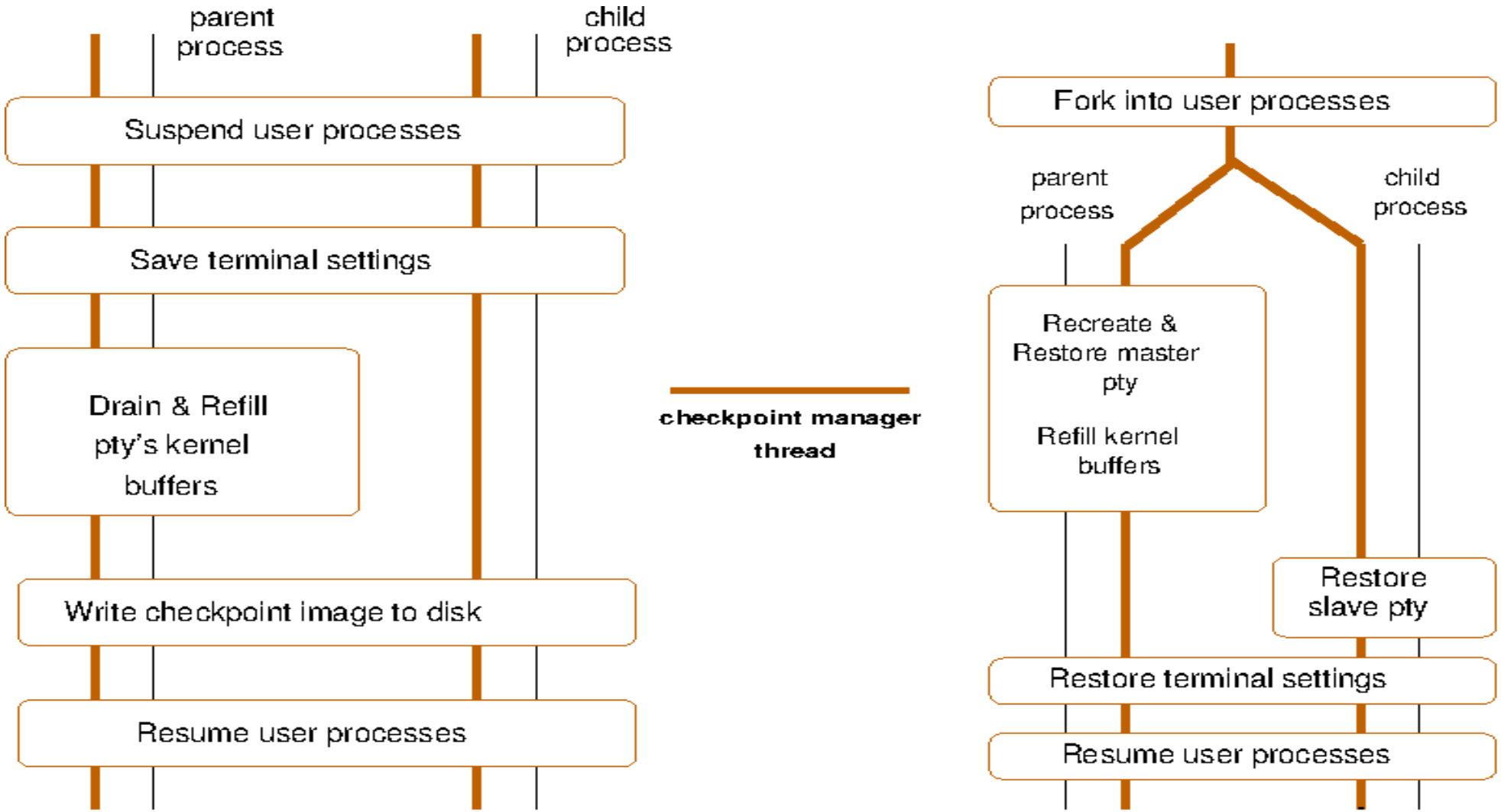


Fig. General approach for a pty-based application

Pseudoterminal plugin

- Wrapper functions for the pty related functions
- Packet or non-packet mode
- Raw or cooked mode
- Terminal settings
- Drain the kernel buffer
- Whether process running in the foreground or background
- Characteristics of bidirectional IPC channel between master and slave devices
- Order of refilling the kernel buffer

Extending DMTCP: Checkpoint & Restart



Live Demo: checkpoint a ROS application

- ROS application details:
  - 3 processes running in 3 different terminal windows
  - roscore
  - Listener node
  - Talker node
- DMTCP demo details:
  - DMTCP coordinator running in the fourth terminal window
  - Manual checkpointing and restart
- Let's start!

DMTCP's growing number of use cases

- Fault tolerance
- Process migration
- Skip past long startup times
- Debugging
- Ultimate bug report
- Speculative execution
- Easily Scalable – over 32,000 nodes checkpointed at ICPADS'16

(Over 50 refereed publications by external teams, describing their use of DMTCP)

DMTCP, ROS2, and the future!

- An RMW(ROS MiddleWare) based on a ROS master and DMTCP
- DMTCP is transparent to the end user because it is built into the RMW
- Combining DMTCP with a small log-and-replay implementation allows rollback and replay during recovery for full reliability.
- This allows an RMW developer to provide a small, reliable RMW, while delegating to DMTCP developers the burden of guaranteeing fault tolerance.
- DMTCP is free and open-source. It uses GNU LGPL (non-contagious): no license fees and can be combined with proprietary corporate software.

Summary

- DMTCP now solves the single point of failure problem of the ROS master.
- Can now checkpoint-restart a typical ROS application.
- Plugin-based architecture helps in extending it easily.
- Can be bundle with larger application package.
- In the future, we hope to work along with upcoming ROS2 framework.

Thank you

This work is also the result of a joint collaboration with Jean-Charles Fabre, Michael Lauer, and Matthieu Amy of the dependable computing and fault tolerance team At LAAS, Toulouse, France.

Appendix: Using DMTCP with ROS

Requires developer version of DMTCP and Linux kernel version above 3.8.0

Execute on <COORD_HOST>:

```
dmtcp_coordinator --port 7779
```

Execute once for each ROS master/node (ckpt interval of 5 seconds):

```
dmtcp_launch --join --interval 5 --port 7779 \
     <path_to_the_ROS_executable>
```

Execute once for each ROS master/node:

```
dmtcp_restart --join --port 7779 --host <COORD_HOST> \
     <path_to_each_checkpoint_file>
```

Appendix: Using DMTCP with ROS (cont.)

- Can change 7779 (the default port) of the DMTCP coordinator to another TCP port, as desired.
- On each ROS node, the checkpoint files will be of the form ckpt_*.dmtcp. Note that ROS sometimes creates additional Python processes, resulting in checkpoint image files for Python. Be sure to include the related Python checkpoint files when restarting the roscore/master.
- Note that if we checkpointed in three different Linux terminals (e.g., master, talker, listener), then we must invoke dmtcp_restart in the three different terminals, with the appropriate ckpt_*.dmtp files for the corresponding ROS node.
- If one prefers manual checkpointing (no '--interval' flag), then one can execute 'c' (checkpoint) in a terminal window running the DMTCP coordinator, optionally followed by 'k' (kill).
- The DMTCP coordinator is stateless. It's fine to kill any old coordinator and start a new one on restart.

References:

1. Ansel, Jason, Kapil Arya, and Gene Cooperman. “DMTCP: Transparent checkpointing for cluster computations and the desktop.” *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 2009
2. Jean-Charles Fabre, Michael Lauer, Matthieu Amy. “Adaptive Fault Tolerance on ROS: A Component-Based Approach” ROSCon 2016.
3. Kerrisk, Michael. *The Linux programming interface*. No Starch Press, 2010.