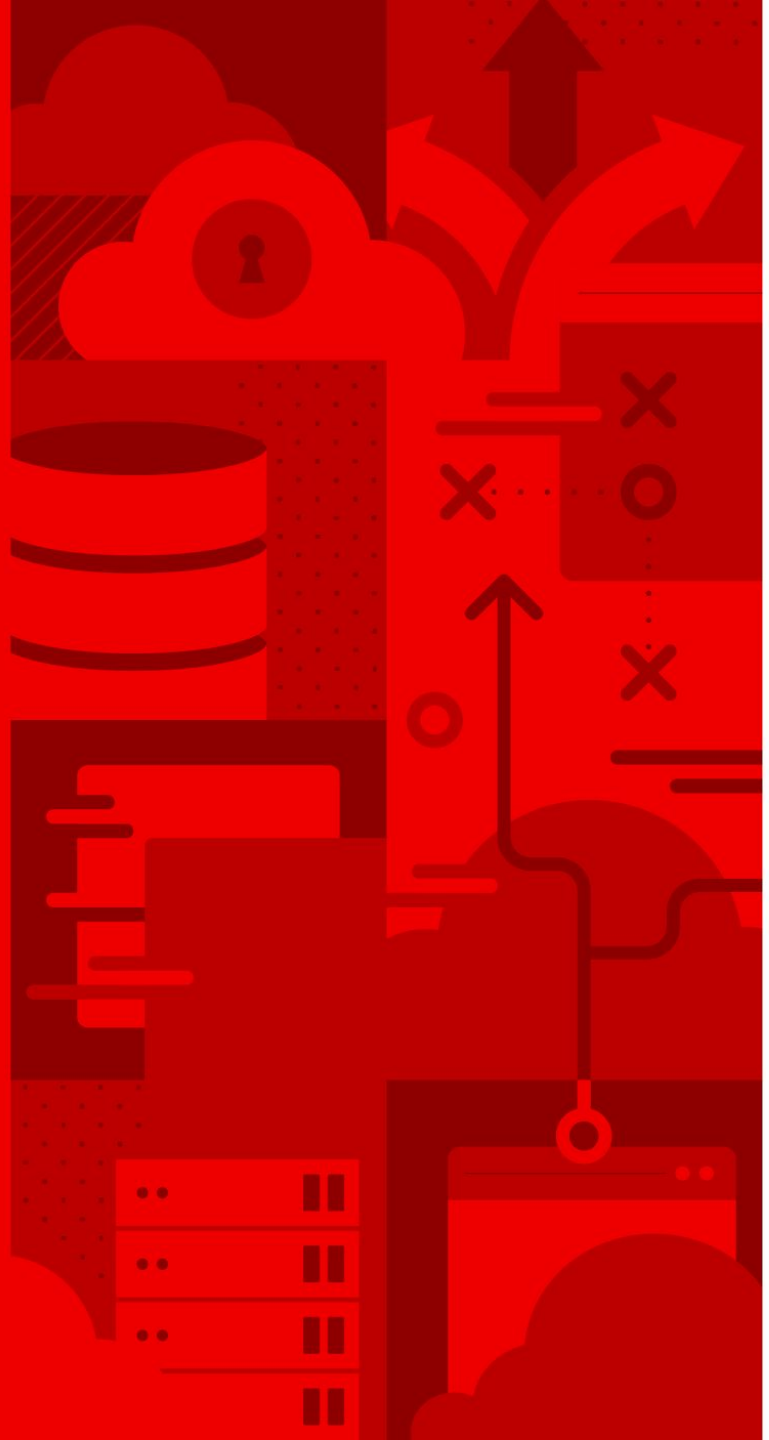

Optional section marker or title

Inclusive language in the network stack and more random stuff

Paolo Abeni

Red Hat

A container for a mostly unrelated topics

- Inclusive language
- Complete refactor of the GRO engine
- IP frags are still a problem
- I still like MPTCP ;)

Slaves no more

- Bonding & bridging have some serious DEI issues... (*)
- Complete stack cleanup is unfeasible
- The worst pain-point is uAPI exposure

(*) I'm the wrong person [to cast] to make this point

A possible way out

- Deprecate the existing problematic uAPI
- Add new enums/defines with inclusive names and same values
- Quite a bit of code churn, but relatively safe
- A redhat colleague already volunteered to have this fun

Problems/issues I don't see??? Alternatives/better approaches?

The sorrows of a GROed packet

- S/W GRO can use quite a few CPU cycles, and even H/W GROed packets experience it
  - Need to parse the packet header, often on cold cacheline[s]
  - The GRO code is scattered in different compilation unit, with sub-optimal locality
  - Need to traverse at least 2 lists, using several additional data cachelines
  - Am I missing some relevant [pain] point ???

Dumb idea, lot of work, small dividends

- Cold cachelines problems are unsolvable (right?)
- All the others problems related to the current GRO engine design: modular/supporting pluggable extensions

What about a monolithic GRO engine? Not entirely monolithic, but at least plain TCP (and UDP?) code could be moved into the core, more fancy stuff will still be modular.

No figures available at this moment.

IP fragmented traffic still causing pain

(at least to me)

- With good parameter the IP defrag stage performs well, don't suffer DoS, nor use excessive memory
- Current defaults are not good
- End-user keep using fragmented traffic no matter what

Time for new defaults?

- Or not, distros can always use their own sysctl setting
  - But does the current one make any sense?
- Just add a lot of MBs to the IP frag cache
  - system with many netns will end OOM
- Different limits enforcing (only global? Both per netns and global?

  What else?)

Wearing the MPTCP hat

- Are there any H/W vendor interested in implementing MPTCP offloads (e.g. checksum, TSO mptcp-aware)
- Is MPTCP blocking some TCP-specific optimization?
  - Or generally speaking, things MPTCP should improve?
- Should (and/or could) MPTCP use PR?
  - Should we clarify/set in stone reqs for subsystems sending PR?

Self-tests code coverage

- How to evaluate it? KCOV needs explicit application logic, per process
- LD_PRELOAD + __contructor__/__destructor__ trick does not works well (awk oops, still investigating why)