CnuasNIC, Datasheet

Virtual RDMA network adapter, RoCEv2 and native InfiniBand

Document DS-CNU-003 • **Revision** C • **Issued** 12 August 2026 • **Status** Published

Item	Value
Part	`cnuas-vnic`
Type	RoCEv2 + native InfiniBand virtual RDMA NIC, guest resident
Version	`v0.1.0-26-gbc3a3c9`
Repo	PacketFive/CnuasNIC

1. Overview

CnuasNIC is the per-VM RDMA network adapter of the Cnuas fabric. The `cnuas-vnic` QEMU PCIe device, together with two kernel drivers and a `libibverbs` provider, presents a fully functional RDMA NIC that speaks both **RoCEv2** and **native InfiniBand** to the CnuasSwitch fabric. Standard verbs applications (perftest, `ibv_rc_pingpong`, MPI, NCCL, UCX) run unmodified against it.

Deployment modes

CnuasNIC is **the one Cnuas fabric component that requires a guest**. This is a deliberate consequence of what it is for, and the distinction matters when planning a deployment:

Component	Runs on the host with no guest
CnuasGPU as a Soft-GPU	Yes
CnuasSwitch	Yes
CnuasLink	Yes
CnuasNIC	**No**

The reason is structural. The value of CnuasNIC is that an **unmodified kernel RDMA stack binds to it**, so the provider is written to the kernel `ib_uverbs` ABI and the drivers bind to an emulated PCI function. Both of those need a kernel with the device enumerated, which is what the guest supplies. A host-only build would have to bypass the kernel stack, and would then no longer be demonstrating the property the device exists to demonstrate.

Two clarifications, because both have been misread:

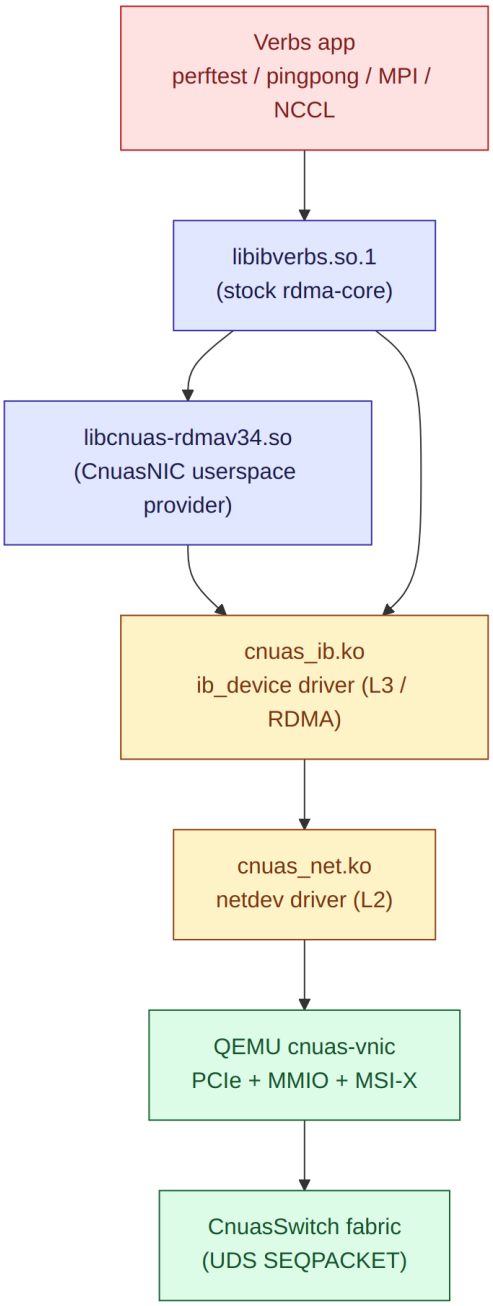
- The `userspace/` "standalone `libcnuas.so`" means the provider **built outside the rdma-core tree**. It does not mean the provider runs without the kernel driver.
- A host process can still speak to a switch port directly, which is how the fabric is tested without virtual machines. That exercises **CnuasSwitch**, not CnuasNIC, and it does not go through the verbs stack.

Key features

- Requires a guest; the drivers bind an emulated PCI function and the provider uses `ib_uverbs`.
- Dual link layer: RoCEv2 (UDP/IPv4 encapsulation) and native InfiniBand (LRH+BTH).
- Reliable Connected (RC), Unreliable Datagram (UD), plus SMI/GSI management QPs.
- Full RDMA verb set: SEND/RECV, RDMA WRITE, RDMA READ, and atomics.

- Shared Receive Queue (SRQ) and InfiniBand multicast (attach/detach).
- Hardware-accurate ICRC compute on TX and validation on RX.
- MAD processing with a Subnet Management Agent (SMA) on QP0 and GSI on QP1.
- Drop-in `libibverbs` provider (`cnuas`, rdmav34 ABI), no app changes.

2. Driver stack



3. Functional specifications

Parameter	Value
PCI vendor : device ID	`0x1AF4 : 0x10F0`
Link layers	RoCEv2 (UDP dst 4791), native InfiniBand (LRH+BTH)

Parameter	Value
Ports per device	1 (dual `port_immutable` personality)
QP types	RC, UD, SMI (QP0), GSI (QP1)
RDMA operations	SEND, RECV, RDMA WRITE, RDMA READ
Atomic operations	CMP_AND_SWP, FETCH_AND_ADD
Receive model	Per-QP receive queue and Shared Receive Queue (SRQ)
Multicast	InfiniBand multicast (`attach_mcast` / `detach_mcast`)
Max path MTU	Up to 4096 B
Integrity	Real ICRC compute on TX, validation on RX
Management	MAD post/recv, SMA GET/SET, directed-route SMPs
Multi-packet	Segmented WRITE/READ honouring negotiated path MTU

4. Kernel driver features

Two kernel modules make up the guest-side device.

4.1 `cnuas_net.ko`, netdev driver (L2)

- Implements `net_device_ops`: `ndo_open`, `ndo_stop`, `ndo_start_xmit`, `ndo_get_stats64`.
- Owns the PCIe device, MMIO doorbell ring, and MSI-X interrupts.
- NAPI receive path; feeds RoCEv2/IB frames to/from the QEMU vNIC over UDS.

4.2 `cnuas_ib.ko`, InfiniBand `ib_device` driver (L3 / RDMA)

Implements the `ib_device_ops` verb set (registered against ib-core):

Category	`ib_device_ops` entry points
Device / port	`query_device`, `query_port`, `query_gid`, `query_pkey`, `get_port_immutable`, `get_link_layer`
Protection domain	`alloc_pd`, `dealloc_pd`
User context	`alloc_ucontext`, `dealloc_ucontext`
Memory region	`reg_user_mr`, `dereg_mr`
Completion queue	`create_cq`, `destroy_cq`, `poll_cq`, `req_notify_cq`
Queue pair	`create_qp`, `modify_qp`, `destroy_qp`
Shared receive queue	`create_srq`, `modify_srq`, `query_srq`, `destroy_srq`, `post_srq_recv`
Address handle	`create_ah`, `create_user_ah`, `destroy_ah`
Data path	`post_send`, `post_recv`
Multicast	`attach_mcast`, `detach_mcast`
Management	`process_mad` (MAD/SMA on QP0/QP1)

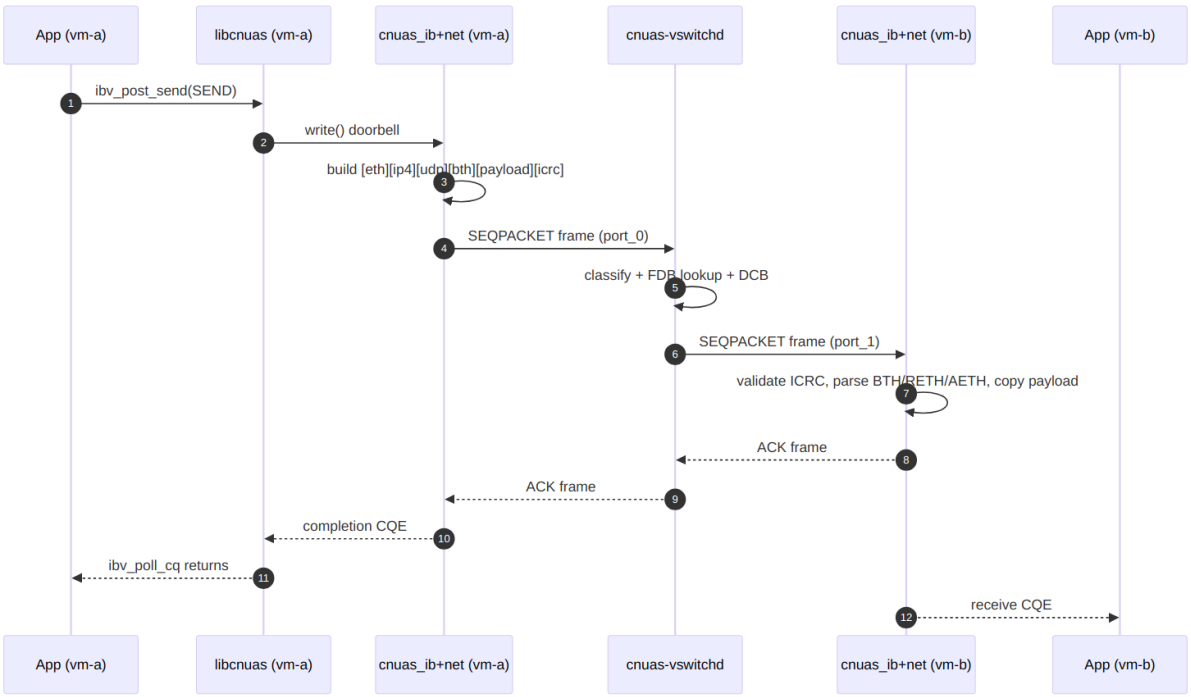
Additional in-driver logic: RC reliability (PSN/ACK/NAK), RDMA READ response generation, atomics execution, ICRC compute/validate, LRH-aware RX dispatch, and SMA attribute handlers.

5. Userspace provider features

- Provider name `cnuas`, against the rdma-core `rdmav34` ABI.
- Two libraries ship: `libcnuas-rdmav34.so`, a drop-in for the rdma-core tree, and a standalone `libcnuas.so` built without it.

- Installed to `/usr/lib/x86_64-linux-gnu/libibverbs/` and `dlopen()`ed by `libibverbs`.
- Implements the userspace `verbs_context_ops` mirror of the kernel verbs above (QP/CQ/SRQ/MR/AH create + post/poll fast paths via mapped doorbells).
- Kernel↔userspace ABI shared through `cnuas-abi.h` (command/response structs).

6. RDMA data flow (RoCEv2 SEND)



7. Interfaces

Interface	Description
Host bus	QEMU PCIe device (`cnuas-vnic`), advertises PCIe Gen5 x16
Fabric transport	UNIX domain socket to CnuasSwitch (shared fabric)
Kernel netdev	`cnuas_net.ko` (`net_device_ops`)
Kernel RDMA	`cnuas_ib.ko` (`ib_device`)
Verbs provider	`cnuas` (rdmav34), `libcnuas-rdmav34.so`
ABI header	`cnuas-abi.h` (kernel ↔ userspace)

8. Verified consumers

`ibv_rc_pingpong`, perftest (`ib_send_bw`/`ib_write_bw`/`ib_read_bw`), `ibstat` / `ibv_devinfo`, and NCCL/UCX build+link gates.

9. Typical usage

```
# In a guest VM with the cnuas-vnic PCIe device
sudo insmod kernel/cnuas_net.ko
sudo insmod kernel/cnuas_ib.ko
ibv_devices        # lists cnuas_0
ibv_devinfo
ib_send_bw         # perftest
```

10. Ordering / integration information

Item	Value
Repo	`PacketFive/CnuasNIC`
Submodule path	`src/cnuasnic` (in `PacketFive/cnuas`)
Version	v0.1.0
Language	C (GNU C, kernel + userspace)
License	GPL-2.0-or-later OR MIT (kernel modules GPL-only)

11. Revision history

Revision	Date	Notes
A	2026-07-05	Initial datasheet
B	2026-07-05	Added driver-stack diagram, full verbs table, kernel/userspace feature detail (through I5f)
C	2026-08-12	Recorded deployment modes and stated that a guest is required, unlike the other fabric components