RDS/TCP: More than just Cluster/IPC.

Sowmini Varadhan(sowmini.varadhan@oracle.com)

Agenda

- What is RDS?
- Motivation for RDS-over-TCP
- Architectural overview of Linux RDS-TCP implementation
- RDS-over-TCP usage in the Cloud
  - RDS/TCP comparisons with other L3 encaps technologies in the Cloud, common challenges, differences
- New features/solutions in the pipeline for RDS:
  - Infrastructural changes in kernel, e.g., Network Namespace support, handling encryption, QoS
  - Protocol changes for addressing security concerns,
  - Deployment issues e.g, Security key distribution

What is RDS?

- “Reliable Datagram Socket”, protocol family PF_RDS: http://linux.die.net/man/7/rds
- Datagram socket that guarantees reliable, ordered, datagram delivery.
- Application sees sendmsg()/recvmsg() datagram semantics. Underlying transport (IB, TCP etc.) takes care of the reliable, ordered delivery.
- Historically, RDS was motivated by the IB use case: in addition to the high bandwidth supported by IB, the native IB queue-pair semantics, QoS, congestion control, RDMA are valuable to RDS applications

Motivation for RDS over TCP

- Initial RDS implementation allowed RDS to use TCP/IP (RDS-TCP) instead of IB to allow functional test verification without requiring IB hardware for test-suite execution.
- As commodity Ethernet chips achieve 40 Gbps (and soon, 100 Gbps), enhancing RDS-TCP to provide relevant QoS etc features over ethernet is extremely attractive for a diverse Distributed Computing Cluster environment
- Cloud and DbaaS offers new potential use-cases for DB apps. Ethernet is the prevalent transport, and RDS-TCP can exploit this potential efficiently

RDS/TCP usage in the Cloud

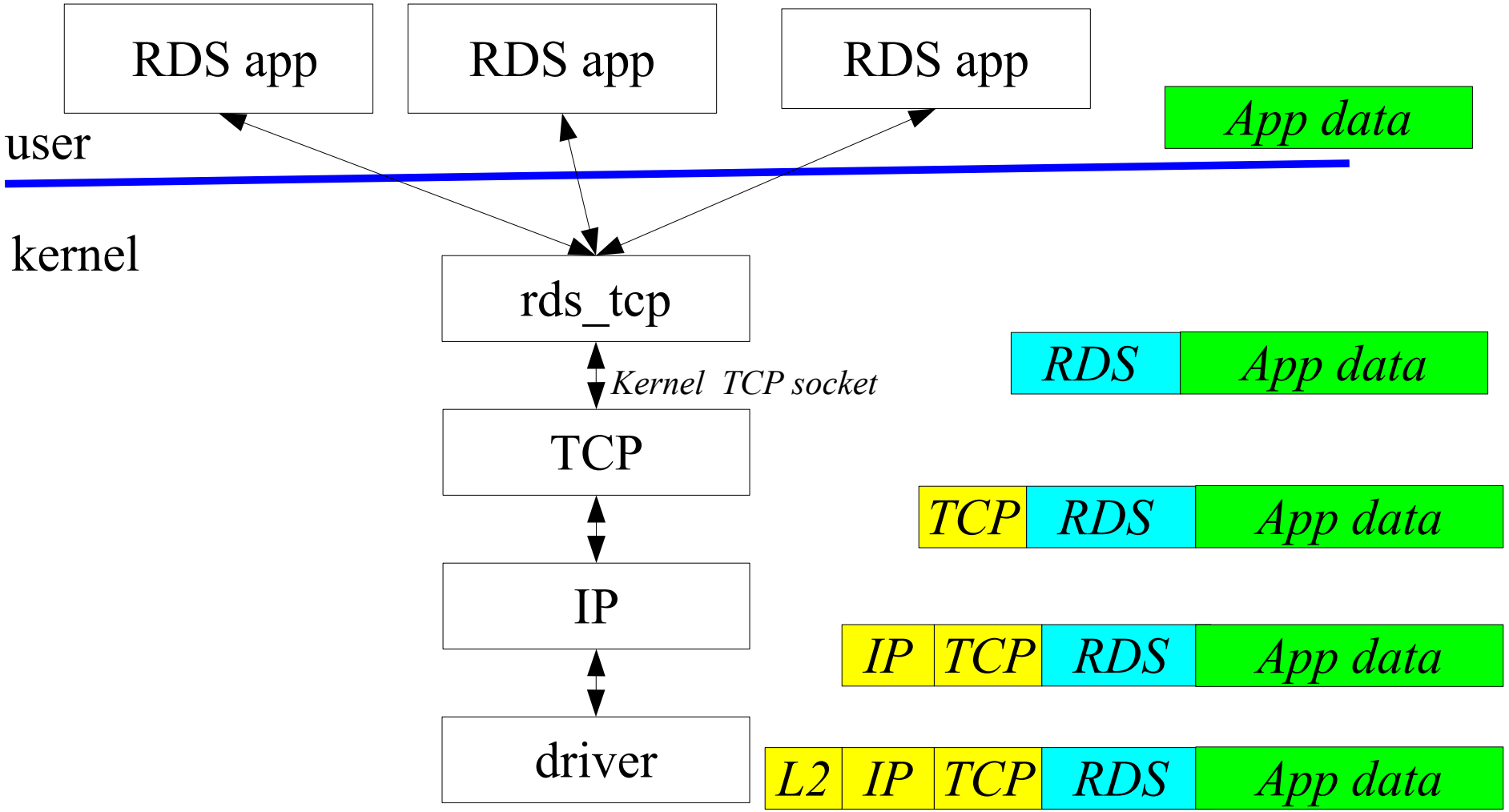
- Cluster applications running in the Cloud need to leverage from the high-speed ethernet-based CLOS networks commonly encountered in Datacenters
  - RDS-TCP can let these RDS apps to run with minimal changes.
  - Feature requirements that arise from this model have some overlap with related L3 encapsulation technologies
- To understand this relationship, the next few slides will give an overview of the RDS architecture

.

What is RDS-TCP?

- RDS datagram is sent over a TCP/IP connection that is managed by the kernel and is transparent to application.
- Application sees sendmsg()/recvmsg() datagram semantics
  - no need to connect()/listen()/accept()
  - Single PF_RDS socket to communicate with multiple peers
  - “Avoids N*N connections by using a single queue-pair or TCP connection for transport”
  - Shared congestion control state for the TCP/IP pipe.

RDS-TCP Architectural Overview



RDS-TCP Linux Implementation Overview of client side:

user

socket(PF_RDS, SOCK_SEQPACKET, 0);
bind(...); / IP address of outgoing interface */*
sendmsg(...);

kernel

rds_sendmsg

conn_create_outgoing:
If necessary, create kernel socket and initialize 3WH to TCP port for RDGS (16385)

Attach RDS header, enqueue the packet for transmission via tcp_sendmsg() after connection handshake is complete

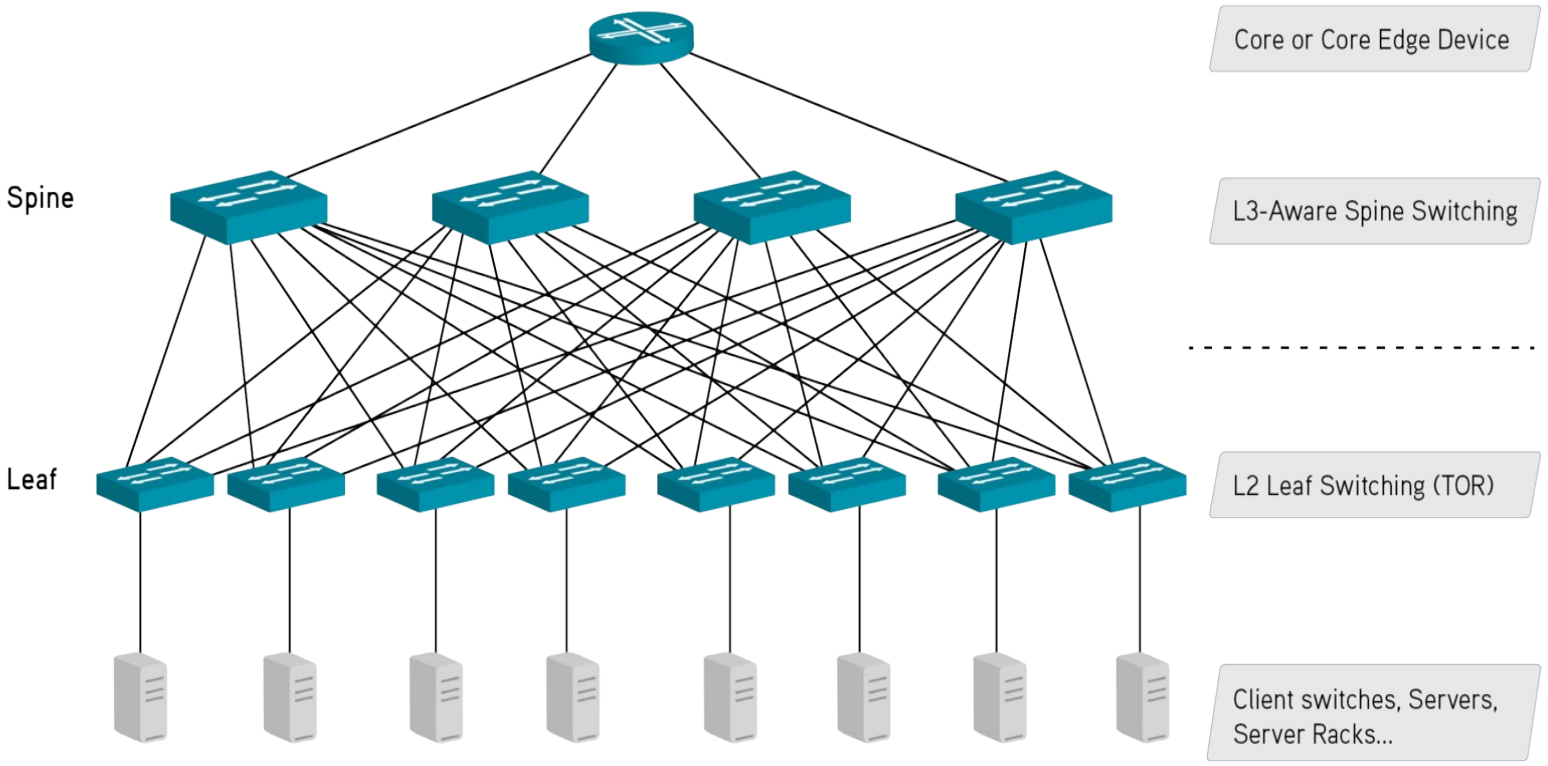
RDS-TCP Linux Implementation Overview of server side:

- 'modprobe rds_tcp' sets up a kernel TCP socket that listens at INADDR_ANY.16385 (RDGS port registered with IANA) */* *some changes in the pipeline for handling network namespaces */*
- user-space server application invokes bind() on the PF_RDS socket for the IP address and port at which the service is to be advertised, and can then invoke recvmsg()
- Kernel rds_tcp_data_ready() is invoked as the →sk_data_ready() callback from TCP, and stitches relevant incoming TCP segments together in rds_tcp_data_recv(). Complete datagram (based on RDS header length) is delivered to application to meet SEQPACKET semantics.

RDS-TCP as an encapsulating method

- Application payload gets encapsulated in an RDS header and tunneled over TCP/IP.
  - Some analogies to VXLAN/GUE etc.
- First a quick tour of the Datacenter problem space, and how VXLAN-like encapsulation techologies are currently being used in this space...

Classic CLOS topology used in datacenters



VXLAN Problem Statement

- Extend VLAN over an IP network
- In a multi-tenant environment, allow each tenant to have the max of 4096 VLANS (vlan field is 12 bits)
- Encapsulate tenant's L2 frame in UDP packet: the UDP source port provides a level of entropy for ECMP/load-balancing in the internal switches
- Other benefits and protocol overview in https://datatracker.ietf.org/doc/rfc7348/

VXLAN Frame format

- Tenant payload is a full L2 frame, with tenant's VLAN
- VXLAN header has a 24 bit VNI (segment identifier)
  - 24 bit VNI X 12 bit tenant VLAN ID
- UDP source port is a hash of “tenant payload” fields

Variants on VXLAN

- STT “Stateless Transport Tunneling Protocol for Network Virtualization”
  - Tenant payload encapsulated in TCP
  - http://tools.ietf.org/id/draft-davie-stt-06.txt
  - Main motivator: leverage from TSO support on NICs
  - Avoids TCP 3WH to be “stateless”
- VXLAN-GPE “VXLAN Generic Protocol Encapsulation”
  - Inner most packet can be an IPv4 or IPv6 frame (not necessarily an L2 frame)
- Others: Geneve (for OVS), GUE (Generic UDP encapsulation)
- Common theme: encapsulation in TCP/UDP

How does this relate to RDS

- Encapsulation at different layers in the stack (RDS encapsulates at the socket layer), protocol specific meta-data header. E.g., RDS vs VXLAN:

- TCP encapsulation, as with STT.
  - the TCP encaps for RDS gives us TSO and even TCP congestion management for free.
  - Individual segments from a datagram do not have the RDS header (impacts entropy for ECMP) – so interior switches are limited on DPI (Deep Packet Inspection)
- Common lessons to be learned, similar challenges to overcome.

Features/challenges for RDS-TCP in the Cloud

- We'll now go over some of the new features we are working on for RDS-TCP deployment in the cloud
  - Privacy and Security Concerns
  - Virtualization
  - QoS
- Each feature has to provide RDS-IB parity where possible, while also addressing the different networking environment of the Cloud.

Privacy and Security Concerns

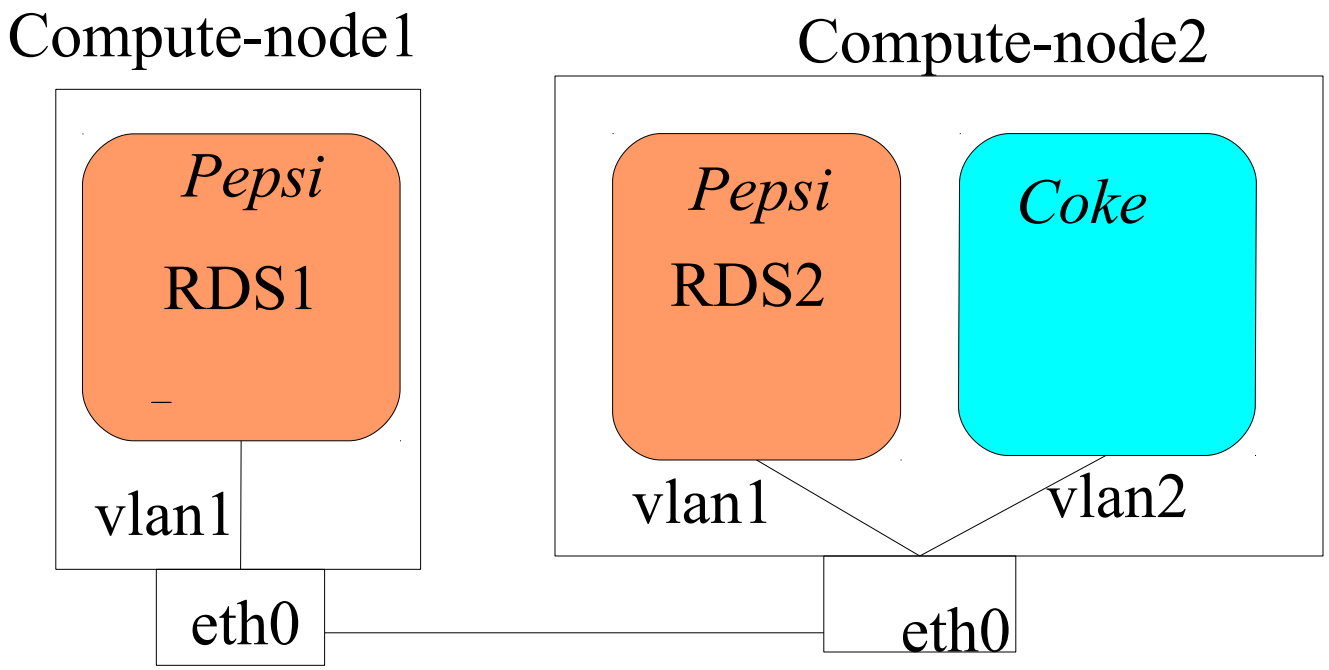
- Cloud Environment presents a new set of security challenges.
- Attack vectors that need to be considered:
  - Protecting Tenant payload (intra-tenant protection)
  - Inter-tenant protection
  - Protecting the control plane (TCP/IP, for RDS-TCP)

Intra-tenant security

- Minimally, RDS payload must be encrypted
- Options to achieve this:
  - Transport level IPsec to the kernel TCP socket. Easy to implement, but,
    - need to orchestrate key distribution
    - Encrypts RDS header as well
  - VPN tunneling: adds extra encapsulation overhead
  - DTLS at the RDS layer
    - Under evaluation.

Inter-tenant traffic separation

- Typical way to achieve this is to use VLANs and VXLANs to separate tenant traffic on ethernet.
- Pepsi traffic on vlan1 cannot be intercepted by any application in the “Coke” Virtual Machine



Infrastructural changes: RDS-TCP and Containers

- Recent commit queued for 4.3: supporting multiple RDS-TCP instances on the same physical machine in separate network name-spaces.
  - rds_tcp registers as a pernet subsys that sets up and tears down kernel sockets on netns creation/deletion
- Will make it possible to run Pepsi and Coke RDS applications in Containers on a single compute-node

Protecting the control plane

- Install ACLs and filters to restrict TCP/IP peering within the cloud.
- Use some form of Authentication to protect the TCP/IP header (TCP-AO, MD5, IPsec Auth?)
  - Key distribution, rollover has to be addressed
- RDS protocol changes:
  - Do not send out RDS connect retries at an aggressive rate! Current specifications (one retry per second, with no backoff!) are based on InfiniBand, but scenario is different for traffic that can traverse a shared ethernet pipe.
  - Protection against spoofing or MitM attacks of RDS port congestion messages

Privacy and Security: features common to all L3 encaps solutions:

- Infrastructure that provides CA and identity verification services for tenants
- Key management services: establishment, management, and secure distribution of cryptographic keys to be used by the control plane
- Securing the Controller itself: ongoing discussions at sdnrg in IETF.

RDS – Entropy issues for ECMP.

- Existing DC encapsulation technologies create the UDP/TCP source port as a hash of the Tenant payload so that the src port can be used by internal switches to do efficient ECMP for a Tenant flow.
  - Could use the ports in the RDS header to achieve something similar.
  - Depending on the choice for encryption, encrypting RDS header means we could lose the entropy
    - Transport mode IPsec: SPI can be used for entropy.
  - RDS header not available in every TCP segment

Advanced features: QoS parity between RDS-IB and RDS-TCP

- Classical RDS-IB uses IB's inherent QoS mechanisms
  - Virtual lane (vlane): independent set of receive and transmit resources associated with a port
  - WRR to select vlane based on service-level marking in the packet
  - 8 bit Traffic Class and 4 bit service level
- For RDS-TCP, equivalent functionality can be achieved using
  - Tc and qdisc at each node,
  - Intserv for resource allocation on the path
  - 6 bit DSCP (Diffserv code-point) + 2 bit ECN, and 802.1p priority (3bit) of the encapsulated packet.

Infrastructural changes for QoS with RDS-TCP

- Specific to RDS: SO_PRIORITY settings on the PF_RDS socket have to be transmitted through so that they can be translated to DSCP/.1p and/or IP_TOS values at the driver/IP layer.
  - Interface with tc to map SO_PRIORITY settings at PF_RDS level to post-encaps qdisc selection
- emulate IB vlane arbitration: dynamically set up tc settings for WRR based on IntServ-based resource allocations along the code path and packet marking for DSCP/.1p
  - Useful for other L3/SDN solutions as well?

RDS-over-TCP Advanced Congestion Avoidance Features

- IB has native support for pro-active congestion avoidance using FECN, BECN with credit allocation
- TCP congestion avoidance is more reactive- it is applied after congestion has already occurred, and is not intrinsically aware of QoS service-level guarantees.
- Provide a controller that monitors service-levels per RDS pair, and pro-actively throttles the sender?
  - Could be generally useful for other encaps techniques as well.

References

- Documentation of the RDS protocol

  https://oss.oracle.com/projects/rds/dist/documentation/rds-3.1-spec.html