NetBricks

Keywords

- Middleboxes
- Network Function Virtualization
- Safe Languages
- Static check Analysis
- Zero Copy software isolation
- Unique types
- vSwitch
- LLVM

MiddleBoxes - Device that performs operations other than the traditional packet forwarding

- Firewall
- Intrusion Detection Systems
- Network Address Translators
- WAN optimizers
- Load Balancers

Network Function Virtualisation

- Offload work of multiple middleboxes to the machines capable of processing different network functions
- Limited use by performance and efficiency

NetBricks - Programming Model - Building NFs

- Abstracted higher level interface to implement network operations
- User Defined Functions

NetBricks - Execution Model - Running NFs

- Zero Copy software Isolation
  - Unique types
- Compile time and runtime checks to enforce memory isolation

Design - Packet Structure

- Stack of Headers
- Payload
- Reference to any per-packet metadata

Design - Programming Abstractions

- Packet Processing
    - Parse
    - Deparse
    - Transform
    - Filter
- ByteStream Processing
    - Window
    - Packetize
- Control Flow
    - Group By
    - Shuffle
    - Merge
- State
    - No external access
    - Bounded inconsistency
    - Strict-consistency
- Scheduled Events

Design - Execution Environment

- Safe Language and runtime environments
- Packet Isolation - call to other NF marks that sender loses access to the packet
- Zero-Copy Software Isolation
- Parallel Directed Graphs for scheduling
- Run-to Completion Scheduling
- Entire NF chain on a single core
- Round robin scheduling policy

Implementation

- Operators running NetBricks chain NFs together and NFs authors use the same language and tools as many optimization can be exploited as a complete program and complex control flow of NFs can be achieved
- Packet processing is lazy as no computation is performed until results are required
- Netbricks process batches of packets at a time for high-performance
- Netbricks uses a modified version of Rust Language and Rust Lint tool developed for static check on unsafe pointers

Costs associated with building Network functions

1. Netbricks NF and equivalent C application performs the same
2. Overheads of array accesses due to bound checking by safe languages

Costs associated with running Network functions

1. Isolation Overheads
    a. Overheads from cache and context switches
    b. Overhead from copying - packet isolation
2. Netbricks performs well due to optimizations on both overheads
3. Netbricks can be executed on multiple cores for better parallelisation
4. As packet processing complexity increases, effect of benefits from netbricks decreases

Check your understanding

- How netbricks provide isolation at low costs?
- Why Stack of headers is required in Netbricks packet structure?
- Why reference of metadata is required in packet structure?
- Implementation of Network functions
- How safe language , static checks and runtime bounds guarantee strong isolation property?
- The behaviour of NetBricks on multiple cores?