NetBricks: Taking the V out of NFV

Scribes

Presented by - Anmol & Sachin

1. **DPDK** - an open source set of libraries and NIC drivers for *fast packet processing*, started by intel, now extended to many more processors.
2. **Click**: Framework for writing network functions using modules. Similar to MPI in the sense that modules are provided for functionalities, if the needed functionality can be implemented using these, well and good, if not, then new modules need to be developed. Problem is that much of the NF authors time goes into optimization of new modules.
3. **Memory isolation:** Needed as NFs from different vendors can't trust each other.
4. **Performance isolation:** Performance of one NF must not affect another's. This is also the reason why NFs can't be implemented as **simple processes** in a traditional OS.
5. Why is per packet metadata needed in the NetBricks per packet structure?

Ans. TCP maintains states which might need to be referenced while packets are being processed.

6. An example of State Abstraction?

Ans.

7. What is reference counting? Why is it being used here? What are the associated tradeoffs?

Ans. This is used for memory management and is analogous to JAVA's garbage collection. Tradeoffs between the two:

a. Reference counting requires the programmer to deal with circularity and reference graphs
b. Issue with garbage collection is when to run it? It may lead to unpredictable performance.

Rust provides reference counted automatic memory management.
Tradeoffs between automatic and manual memory management:

a. **Program complexity:** Automatic management makes program less prone to bugs (and less verbose too) by taking away control from programmer
b. **Performance:** Manual management is better for performance as it allows more control.

8. Safe type casting examples?

Ans. 1. Must not be able to type cast integer to pointer.

2. Unsigned to signed int conversions need to be checked. Example:

```
Void copy(char *src, char *dst, unsigned len){
        while(len>0)
                *dst++=*src, len--;
}
```

Usage: copy(x,y,**-1**) -> -1 is converted from signed to unsigned and becomes 2^{32}-1, allowing the user to access (potentially) memory that was to be isolated and even crash the system.

More vulnerabilities exist with dereferencing of NULL pointers,, which usually cause the system to invoke system handlers. If a hacker has access to the system such that (s)he can write a custom system handler to invoke malicious behaviour through the program.

9. **Unique types:** It is a stronger type system. The tradeoffs are:

a. User may have to write larger code (code becomes more verbose)
b. There is more burden on compiler to optimize code
c. It provides immutable types, which lessen the potential number of bugs in code

10. Why run-to-completion scheduling?

Ans. In order to try and exploit better locality.

11. What is ***lazy*** evaluation? Why is packet processing abstractions evaluation ***lazy***?
Ans. let x=f(y) be an assignment in some code. If it were an eager language (say C++), this would be calculated immediately and stored. If it were a lazy language (say Haskell or Spark), it will book keep a computational structure (mappings), and x would be evaluated when needed, otherwise it won't be.

Tradeoffs are:

a. Lazy languages need to have stronger understanding of data dependency
b. Need more book-keeping at runtime (may given poorer performance)

Why is it being used here?
-> Makes scheduling of packet batches easy.
12. NF chains can be thought of as a pipeline. Thus, for scalability, NF chains are replicated onto cores.
13. SoftNIC is a virtual switch optimized for NF use-cases.