Sarah Lehman - CIS 5590, Fall 2016

Not Another Network Controller!

Extending Existing Operating System Functionality to Manage Networks

Topics

- Problems with current SDN patterns
- Management capabilities of operating systems
- Applying OS principles to SDN
- Evaluations and discussions

Topics

- Problems with current SDN patterns
- Management capabilities of operating systems
- Applying OS principles to SDN
- Evaluations and discussions

The Current State of SDN

- **Goals**:
  - Centralized control
  - Abstraction and programmability
  - Virtualization

The Current State of SDN

- **Goals**:
    - Centralized control
    - Abstraction and programmability
    - Virtualization

- **Reality**: control is centralized but too complex and inflexible
    - Applications run as part of the framework
    - Tightly coupled; bugs in one area affect whole system

The Current State of SDN

- **Goals**:
  - ~~Centralized control~~
  - Abstraction and programmability
  - Virtualization

- **Reality**: abstractions are available, but are too domain-specific
  - Modules written in mandated languages
  - Frameworks only support certain capabilities

The Current State of SDN

- **Goals**:
  - ~~Centralized control~~
  - ~~Abstraction and programmability~~
  - Virtualization

- **Reality**: achievable but usually with great effort and complexity
  - Resource-heavy translation from physical to virtual
  - Still requires admin to be heavily involved

Introducing YANC

- **Goals**:
  - Applications should encompass logically distinct tasks.
  - Applications may be written in any language.
  - Applications should come from multiple sources.
  - Applications should be decoupled from hardware.
  - The interaction between applications should be defined by the administrator.
  - Network application design should not be limited by the controller.

Introducing YANC

- **Goals**:
  - Applications should encompass logically distinct tasks.
  - Applications may be written in any language.
  - Applications should come from multiple sources.
  - Applications should be decoupled from hardware.
  - The interaction between applications should be defined by the administrator.
  - Network application design should not be limited by the controller.

Introducing YANC

- **Goals**:
  - ~~Applications should encompass logically distinct tasks.~~
  - Applications may be written in any language.
  - Applications should come from multiple sources.
  - Applications should be decoupled from hardware.
  - The interaction between applications should be defined by the administrator.
  - Network application design should not be limited by the controller.

Introducing YANC

- **Goals**:
  - ~~Applications should encompass logically distinct tasks.~~
  - ~~Applications may be written in any language.~~
  - Applications should come from multiple sources.
  - Applications should be decoupled from hardware.
  - The interaction between applications should be defined by the administrator.
  - Network application design should not be limited by the controller.

Introducing YANC

- **Goals**:
  - ~~Applications should encompass logically distinct tasks.~~
  - ~~Applications may be written in any language.~~
  - ~~Applications should come from multiple sources.~~
  - Applications should be decoupled from hardware.
  - The interaction between applications should be defined by the administrator.
  - Network application design should not be limited by the controller.

Introducing YANC

- **Goals**:
  - ~~Applications should encompass logically distinct tasks.~~
  - ~~Applications may be written in any language.~~
  - ~~Applications should come from multiple sources.~~
  - ~~Applications should be decoupled from hardware.~~
  - The interaction between applications should be defined by the administrator.
  - Network application design should not be limited by the controller.

Introducing YANC

- **Goals**:
  - ~~Applications should encompass logically distinct tasks.~~
  - ~~Applications may be written in any language.~~
  - ~~Applications should come from multiple sources.~~
  - ~~Applications should be decoupled from hardware.~~
  - ~~The interaction between applications should be defined by the administrator.~~
  - Network application design should not be limited by the controller.

Topics

- ~~Problems with current SDN patterns~~
- Management capabilities of operating systems
- Applying OS principles to SDN
- Evaluations and discussions

YANC’s Solution

- **How to achieve these goals**?
  Extend a modern operating system to manage networks!
  - Leverage the OS’s file system to represent network state
  - Use the OS’s existing permissions framework to control network security and virtualization

Strengths of a File System

- Files provide a common interface for applications
  - Easy-to-read text-based content
  - Structure and naming conventions provide meaning
- Existing command-line operations can be utilized
  - Every operation updates the network: `echo(), mkdir(), rmdir(), rename()`

Strengths of an Operating System

- File system is built-in, but separate from network infrastructure
    - Underlying drivers, frameworks, etc. can be maintained independently
- Standard permissions used to limit access to files and directories
- Namespaces can be used to slice / virtualize network resources

Topics

- ~~Problems with current SDN patterns~~
- ~~Management capabilities of operating systems~~
- Applying OS principles to SDN
- Evaluations and discussions

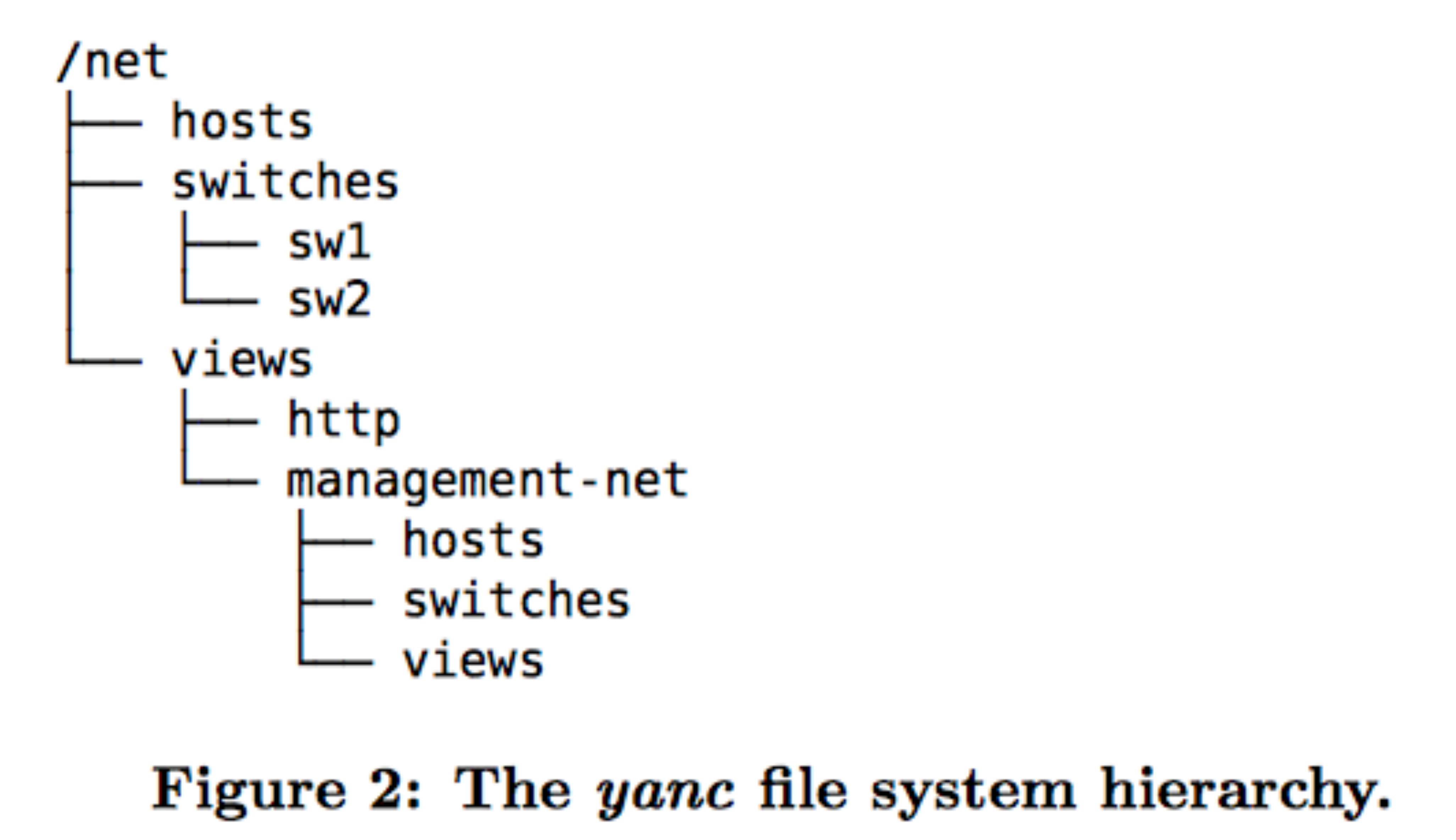
Needs of a Network

- Manage network state and traffic patterns
- Manage access / communications of applications
- Manage user / traffic security
- Manage overall reliability, availability, consistency

YANC's Approach

- Manage network state using the OS file system
  - Represent "coarse" entities as directories
  - Represent "fine" details as files
  - Represent network events as file CRUD events
  - Use file name conventions to represent properties (such as flow matching fields)

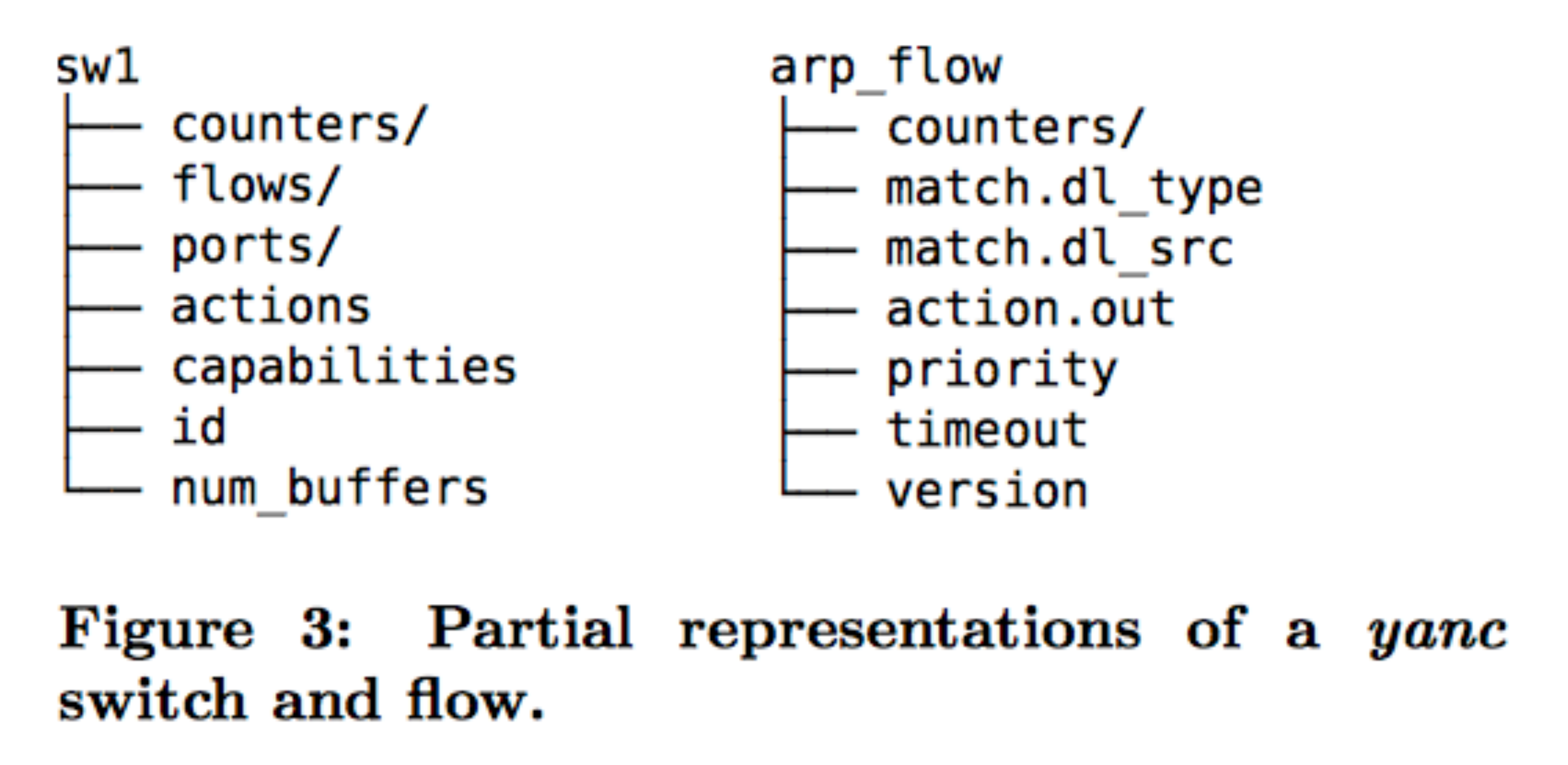
YANC’s Approach



```
/net
├── hosts
├── switches
│   ├── sw1
│   └── sw2
└── views
    ├── http
    └── management-net
        ├── hosts
        ├── switches
        └── views
```

Figure 2: The *yanc* file system hierarchy.

YANC's Approach



```
sw1
├── counters/
├── flows/
├── ports/
├── actions
├── capabilities
├── id
└── num_buffers
```

```
arp_flow
├── counters/
├── match.dl_type
├── match.dl_src
├── action.out
├── priority
├── timeout
└── version
```

Figure 3: Partial representations of a *yanc* switch and flow.

YANC’s Approach

- Manage network security using existing OS features
  - Distributed file systems to create distributed “controllers”
  - Standard permissions to manage app access to files and directories
  - “View” directories and OS namespaces to virtualize network topologies and resources

Topics

- ~~Problems with current SDN patterns~~
- ~~Management capabilities of operating systems~~
- ~~Applying OS principles to SDN~~
- Evaluations and discussions

Performance

- Familiar interface, at a cost
  - File system interactions require context switches
    - Ex: `read(), write(), stat()`
  - Switching context many times introduces overhead
- Future work planned to reduce overhead
  - Very few additional details provided

Future Work

- Limited support for flow table misses / writes in multi-process system
- libYANC - network library to improve performance of flow entry updates and transfer of bulk packets
- Expanding YANC to be directly usable by network devices

Discussions

- System provides limited functionality on its own
  - Requires dedicated applications for topology discovery, path determination, and switch-flow writing
  - No explicit reporting interface
- Operating system **very open-ended** with no accountability
  - Does file system reflect actual network state?
  - How to handle consistency and conflict resolution?

Discussions

- More information required on evaluations and performance
    - Need details on exact tests completed and metrics used
    - No hard figures on actual overhead introduced
    - No remarks on scalability or data storage requirements

Final Remarks

- YANC takes great steps toward providing a level playing field for network administrators and application developers alike.
- Concepts introduced here have merit, even if the prototype itself still needs work.
- Advice to the authors - invest in more rigorous testing and evaluation, and providing accountability for applications.

Questions?

References

- Matthew Monaco, Oliver Michel, and Eric Keller. 2013. Applying operating system principles to SDN controller design. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks* (HotNets-XII). ACM, New York, NY, USA. Article 2, 7 pages. DOI=http://dx.doi.org/10.1145/2535771.2535789