# Accountable SDNs for Cyber Resiliency

## UIUC/R2 Monthly Group Meeting

Presented by Ben Ujcich

March 31, 2017

# Outline

- Motivation for accountability
- Our accepted paper:
  **"Towards an Accountable Software-Defined Networking Architecture."**
  B. E. Ujcich, A. Miller, A. Bates, and W. H. Sanders. *Proceedings of the 3rd IEEE Conference on Network Softwarization (NetSoft 2017)*, July 3-7, 2017.
- Early stage work on implementing SDN accountability
- Ideas and feedback

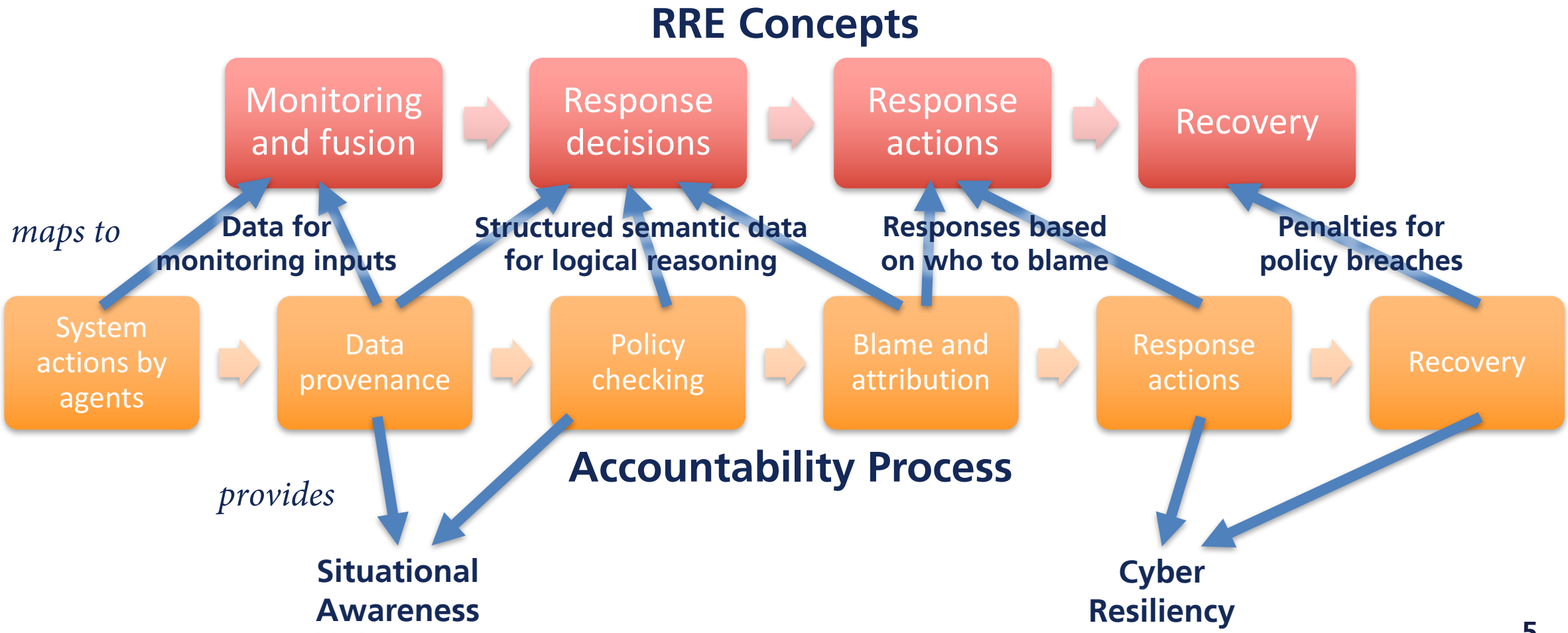
# Motivation

- SDNs provide flexibility, but also new opportunities for attacks
- What assurances do we have about previous system events?
- NIST definition of **accountability**: *"actions of an [agent] [that can] be traced uniquely to that [agent]"* that supports *"nonrepudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action"*
- Why accountability in SDN?
  - **Attributing causal actions** is difficult; needed for **assigning blame** fairly and to take appropriate **response actions**
  - Multiple (potentially distrusting) parties or **agents** with different incentives

# Uses of Accountability

- *A posteriori* compliance control
  - Collect **relevant** data about agents' actions in order to **blame** one or more agents based on **agreed-upon (*a priori*) policies**
- Forensics
  - Collect **all** data about agents' actions under **adversarial conditions** in order to **blame** one or more agents
- Troubleshooting
  - Collect **relevant/all** data about agents' actions for **testing** or **debugging** purposes under **non-adversarial** (but possibly faulty) conditions

# RRE, Accountability, and Cyber Resiliency



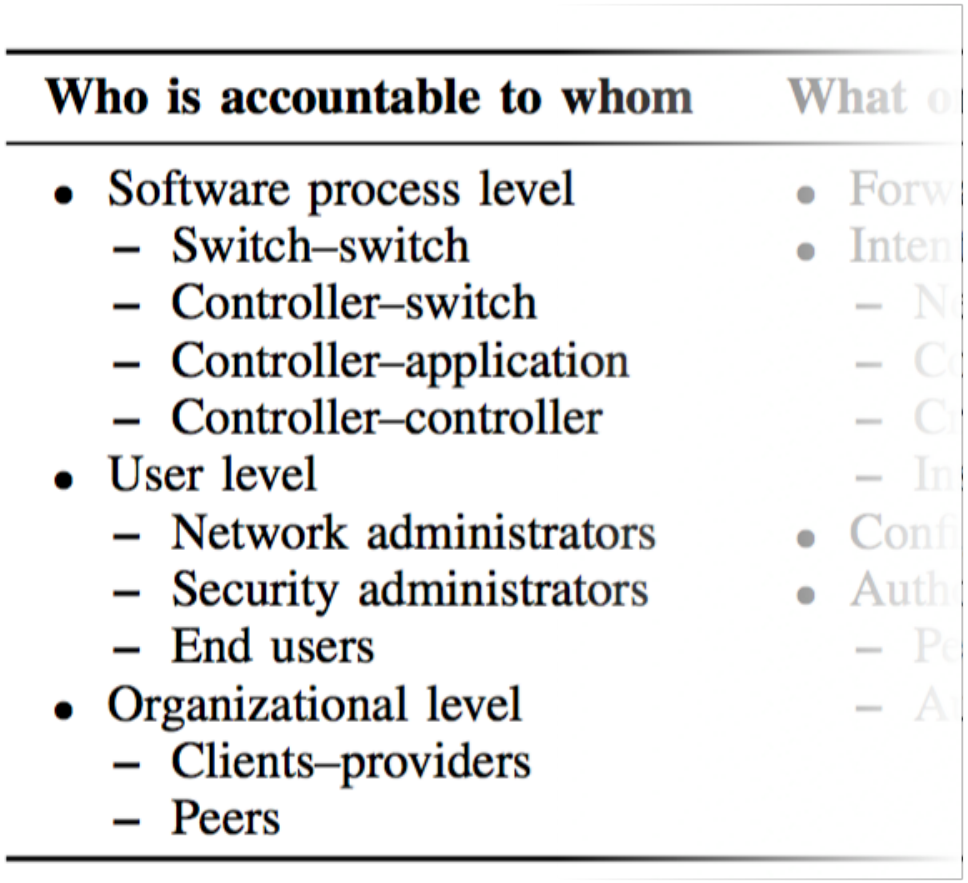
# Approach

- Applying "accountability regime" design[1] to SDNs based on CS and social science notions of accountability

| Who is accountable to whom | What one is accountable for | Assurance mechanisms | Standards | Effects of breach |
|---|---|---|---|---|
| • Software process level<br>– Switch–switch<br>– Controller–switch<br>– Controller–application<br>– Controller–controller<br>• User level<br>– Network administrators<br>– Security administrators<br>– End users<br>• Organizational level<br>– Clients–providers<br>– Peers | • Forwarding / topology<br>• Intent / policy<br>– Network resources<br>– Constraints<br>– Criteria<br>– Instructions<br>• Configuration<br>• Authorization / access<br>– Permissions and roles<br>– Authentication and access | • Data provenance<br>• Authenticated logging<br>– Tamper-proof<br>– Non-repudiable<br>• Fault tolerance<br>– Byzantine fault tolerance<br>– Graphical modeling<br>– Blockchains<br>• Roots of trust | • Legal<br>• Regulatory<br>• Policy<br>• Contractual | • Deterrence<br>– Loss of money<br>– Loss of reputation<br>• Resiliency<br>– Response<br>– Recovery |

[1] J. L. Mashaw, "Accountability and institutional design: Some thoughts on the grammar of governance," in *Public Accountability: Designs, Dilemmas, and Experience*, M. W. Dowdle, Ed., 2006.

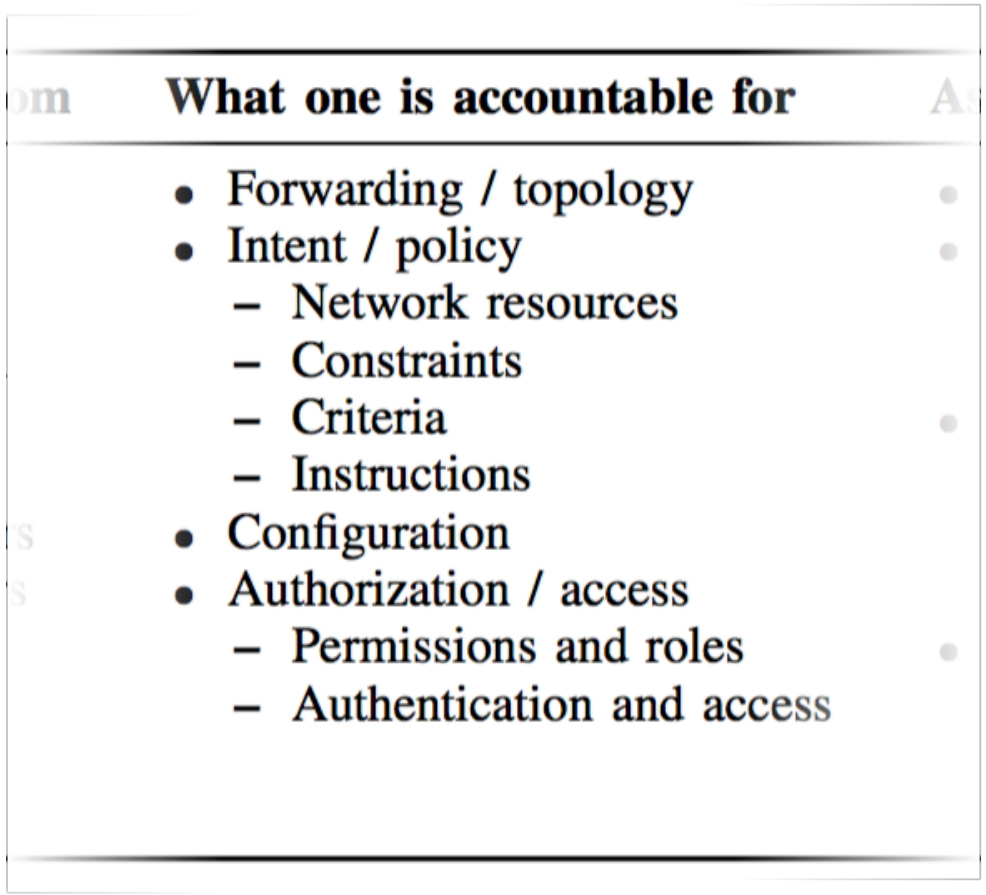
# 1. Who is accountable to whom



| Who is accountable to whom | What |
|---|---|
| • Software process level<br>– Switch–switch<br>– Controller–switch<br>– Controller–application<br>– Controller–controller<br>• User level<br>– Network administrators<br>– Security administrators<br>– End users<br>• Organizational level<br>– Clients–providers<br>– Peers | • Forw<br>• Inter<br>– N<br>– C<br>– C<br>– In<br>• Conf<br>• Auth<br>– P<br>– A |

- Notion of **agents** and their relationships among each other
- SDN "ecosystem" encompasses many interrelated agents that requires looking at the system holistically

# 2. What one is accountable for

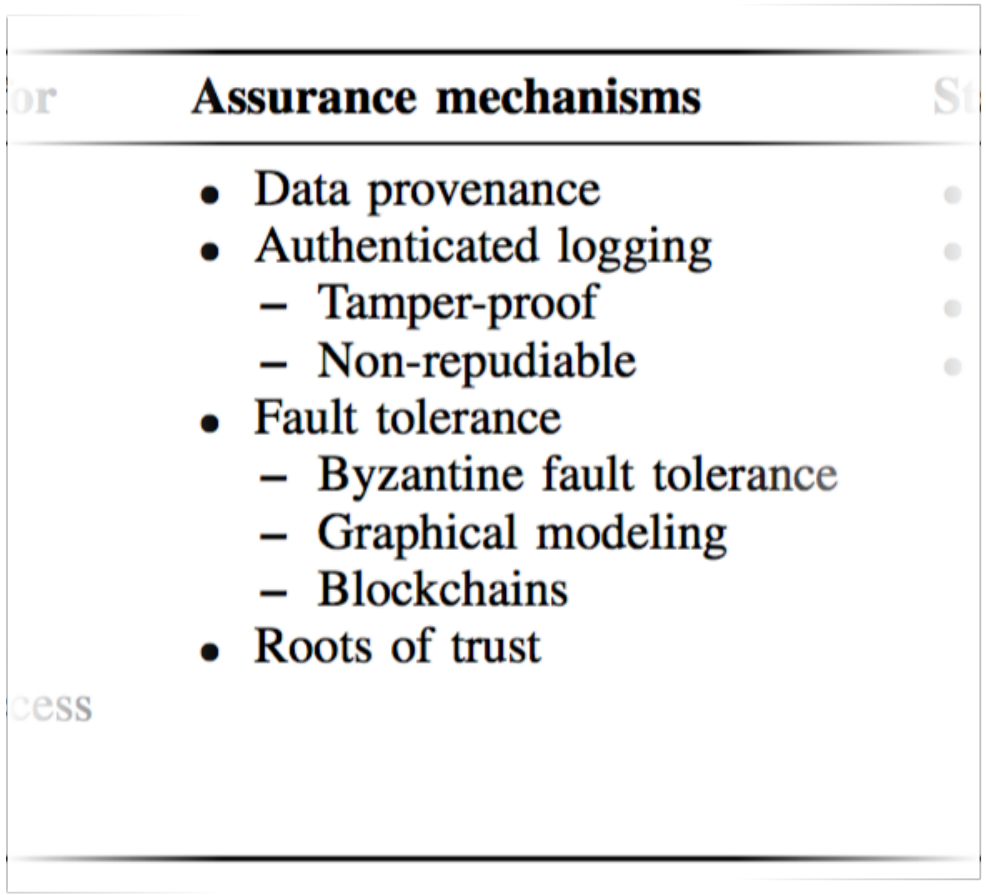


**What one is accountable for**

- Forwarding / topology
- Intent / policy
  - Network resources
  - Constraints
  - Criteria
  - Instructions
- Configuration
- Authorization / access
  - Permissions and roles
  - Authentication and access

- Notions of **entities** that store system state and **actions** that can be taken on the entities by the **agents**
- Broader view than just simply a representation of the network as a forwarding graph

# 3. Assurance mechanisms

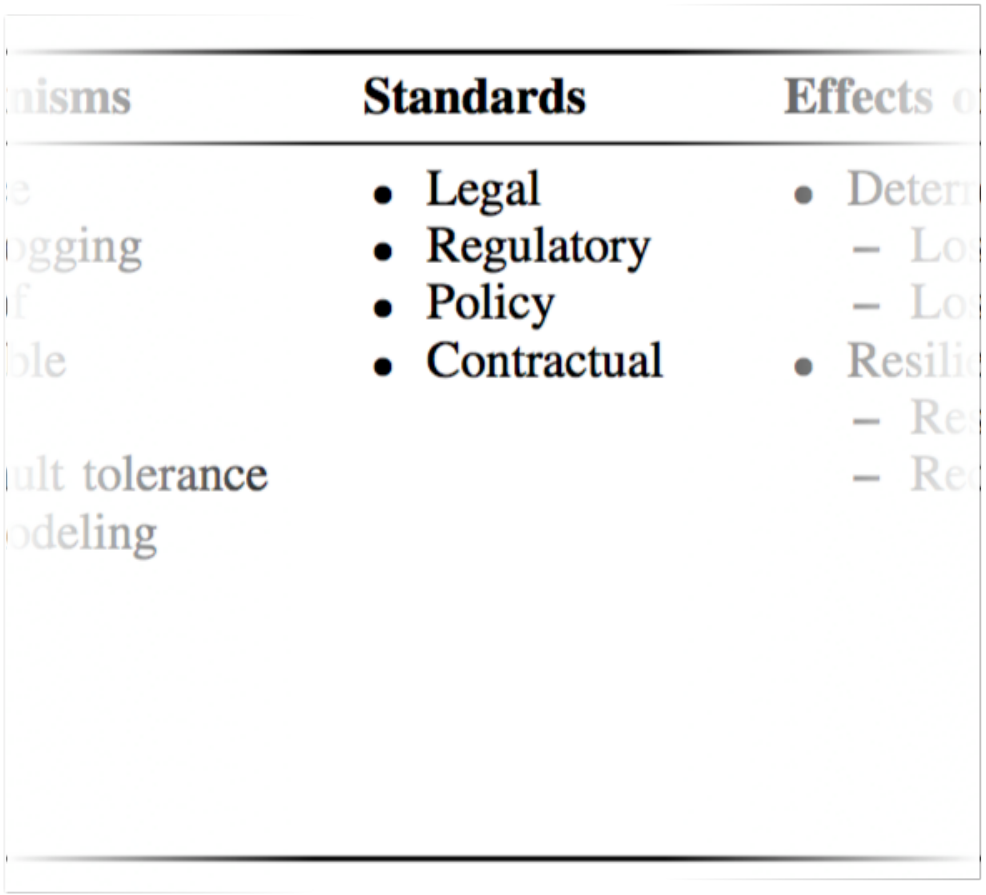


**Assurance mechanisms**

- Data provenance
- Authenticated logging
  - Tamper-proof
  - Non-repudiable
- Fault tolerance
  - Byzantine fault tolerance
  - Graphical modeling
  - Blockchains
- Roots of trust

- What assurances or guarantees can we make about the data that we collect?
- Important research areas:
  - Data provenance
  - Blockchains and cryptocurrencies

# 4. Standards



| nisms | Standards | Effects |
|---|---|---|
| gging<br>ole<br>ult tolerance<br>odeling | • Legal<br>• Regulatory<br>• Policy<br>• Contractual | • Deter<br>– Lo<br>– Lo<br>• Resili<br>– Re<br>– Re |

- Two views of accountability standards
  - Accountability by design to support (external) legal systems
  - Accountability by design to create a system for self-executing policy/compliance enforcement
- Automated enforcement of standards via smart contracts

# 5. Effects of breach

| ndards | Effects of breach |
|---|---|
| Legal<br>Regulatory<br>Policy<br>Contractual | • Deterrence<br>– Loss of money<br>– Loss of reputation<br>• Resiliency<br>– Response<br>– Recovery |

- Go beyond just collecting data for auditing; must use it somehow
- Deterrence and resiliency as complementary aspects
- Completes the RRE "loop"

# Implementing Accountability

- Goal: Design and build a realized accountable SDN system
- Major components:
  - **Data provenance / provenance language** for formally describing system state (i.e., how data came to be) in a structured way
  - **Blockchains** as replicated, fault tolerant distributed consensus ledgers to store commitments about past data provenance
  - **Smart contracts** to implement *a priori* policy agreements among (distrusting) agents for meeting system invariants/predicates and for defining consequences if invariants are breached

# Components: Data Provenance

- RDF triples for building distributed provenance graph
- Ontology constrains language
- **Extend W3C PROV ontology with SDN semantics**
- **Use provenance data model to form queries with networking/security semantics**
  - E.g., "Was there a path between hosts A and B at this time?"

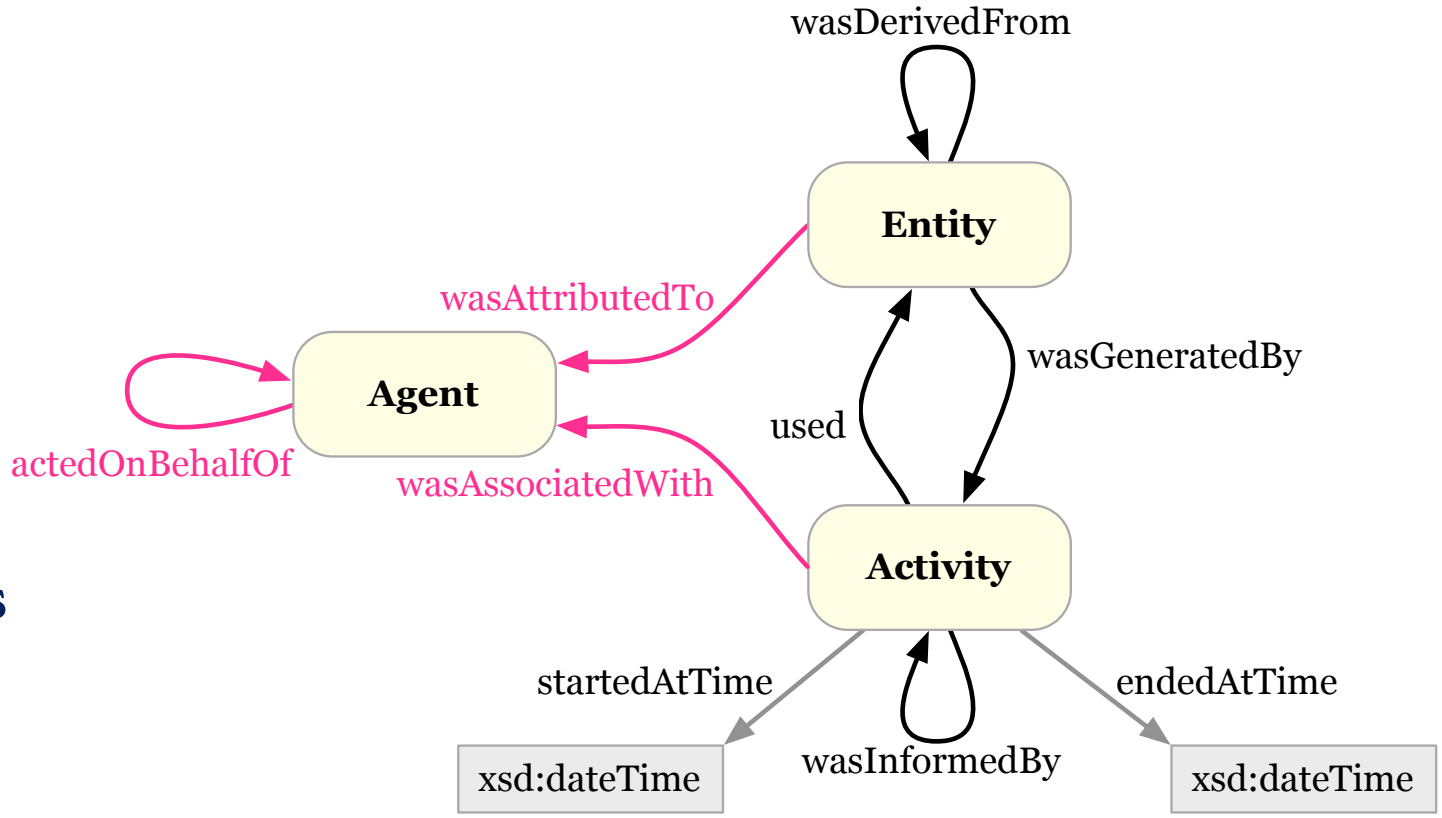



**Diagram source:** "A Walk Through PROV-O", Tim Lebo, 2012. URL: https://www.w3.org/2011/prov/wiki/ISWCProvTutorial

# Components: Blockchains

- Blockchain data stored and executed by all participating nodes for fault-tolerance and replication purposes
- Can store self-executing smart contracts
- Consensus through proof-of-work or BFT-like protocols
- **Agents commit hashes of provenance data to blockchain's smart contracts**
- **Auditing protocol**

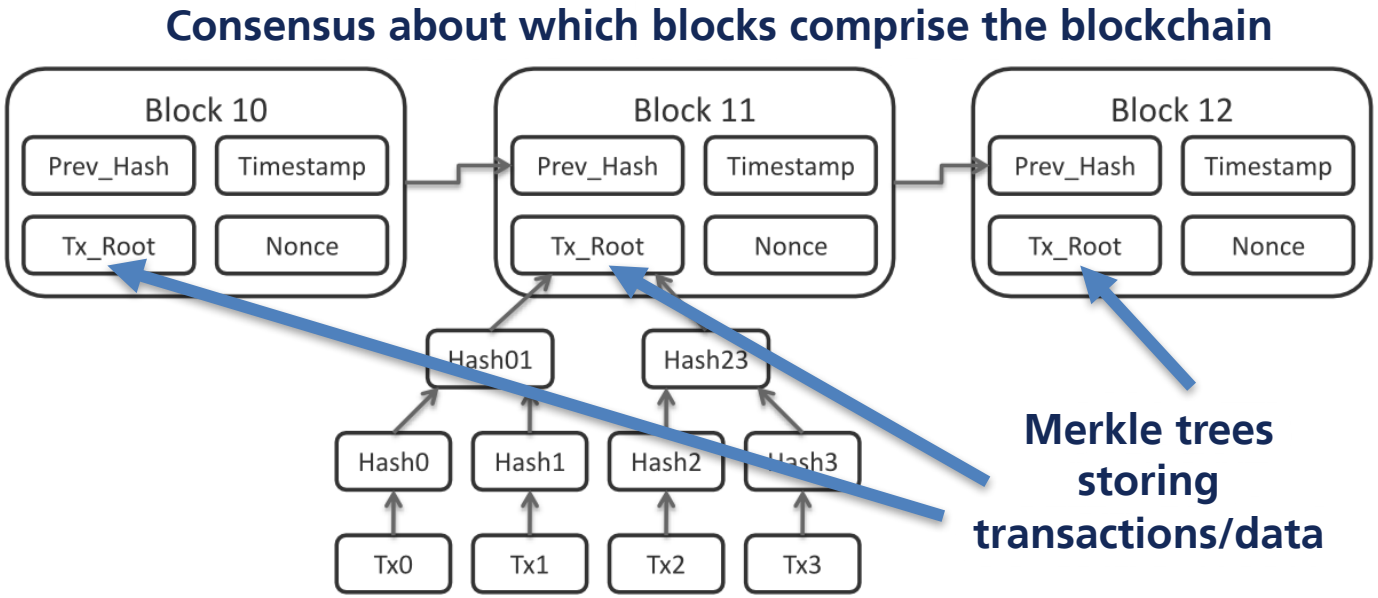
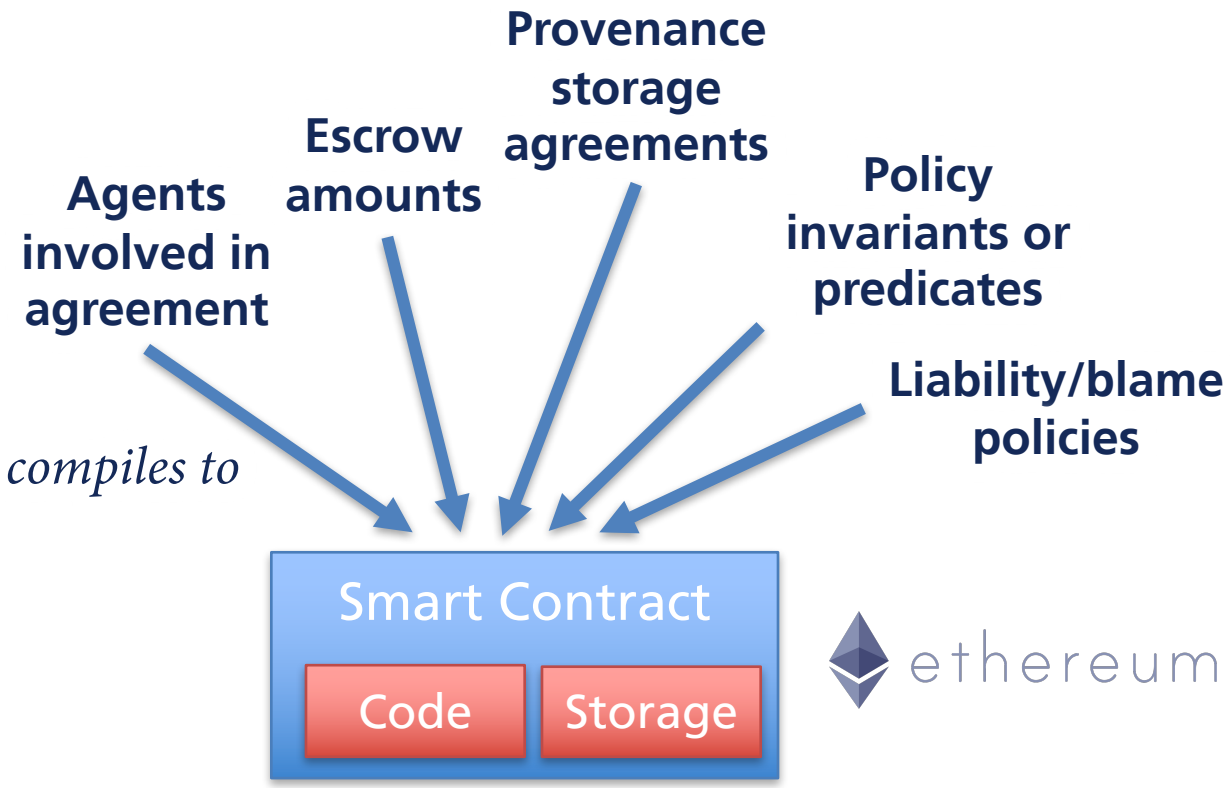



**Diagram source:** "Bitcoin Block Data", Matthäus Wander, Wikimedia Commons. URL: https://commons.wikimedia.org/wiki/File:Bitcoin_Block_Data.png

# Components: Smart Contracts

- Self-executing pieces of code and data that "live" on the blockchain
- In cryptocurrency context, can exchange financial value
- **Store policy agreements among agents and relevant commitments related to provenance**
- **Translate high level network policies to executable code**



# Proposed Case Studies

- Multiple administrative domains
  - Different administrators
  - Different ownership/trust assumptions of equipment, processes, or data
- Network applications
  - Extensions to SDN controller functionality for providing services (e.g., IDSes, firewalls)
  - Proliferation of network applications makes it challenging to assign blame, especially with apps of equal permission levels

# Ideas and Feedback

- Alignment with research goals
- Uses of accountable systems or networks with highly granular provenance metadata and/or automated penalties and responses
- Extension to end host application semantics
- Questions?

# Thanks!