Distributed Name Rewriting

(DINRG Feb 17, 2018, San Diego)

<christian.tschudin@unibas.ch> University of Basel

Overview

What is the common abstraction behind “distributed Internet infra-structures”?

Hypothesis: **Distributed Name Rewriting**

This talk revisits (in 15 min): DNS, BTC, ARP, DHT, NFN, Tangle, GitHub …

Other possible names (instead of DNR) from this morning:
- *distributed secure mappings (bind identities to keys), potentially trustless*
- *NFaaS (securely map inputs to computation results)*

DNS vs BTC

DNS:

`lookup(in) --> out`

e.g. *'in'* is hierarchical domain name, *'out'* is IP number

BTC:

`lookup(in) --> out`

'in' is random (account or tx) number, *'out'* is an account balance, a transaction, a smart contract, an insurance policy etc

DNS vs BTC (contd)

At the end of the day, DNS and BTC both are:

- small to midsize databases (BTC is a ledger, after all)
- global (distributed or replicated)
- simply query interface, maps one name to another

There are differences, of course …

DNS vs BTC - somehow different

	DNS	BTC
versioning (history)	no	yes (IOTA has snapshots and forgets)
consistency	eventually cons. (dependent on caching params)	strong cons. IFF you are on the winning branch
input names (how to prevent conflicts)	unique because pre-coordinated	unique because random
decentralized storage	yes (iterative/recursive remote query)	no (full replica)

one very big difference, though:

DNS vs BTC - really different

b/c of the UPDATE method

DNS: "pre-established agreement on delegation”

updates only possible in delegated subtree, are independent and can be done in parallel

BTC: trustless process

Byzantine Agreement Protocol for global, synchronous consensus

Is “distributed name rewriting" still a good common abstraction? I think yes.

More Name-Rewriting Infrastructures

Seen so far: DNS, BlockChain (BTC), Tangle (IOTA)

ARP - dynamic mapping

Forwarding - routing table with next-hop lookup

DHT - an index, beside DNS the other “exemplary lookup” infrastructure

PKI - `secured_lookup(some_public_key) -> signing_key`

cloud computation - `lookup( fct(in) ) -> result`
Web pages are computation results: lookup results are cacheable, see memcachd

NFN (Named-Function-Networking) — `resolve( symbolic_expr ) -> result`
scalable! immutable inputs, *confluence of resolution strategies avoids need for consensus finding*

Again: “update” is probably the strongest differentiator

Communication is Computation is Distributed Name Rewriting is Communcation is ...

Notation used: `A(something)` means: "something is on host A" `config[..]` represents global state

Story: We want to replicate an item, send a unicast datagram from A to B via X

```
config[ A(srcA,nameB,item), B(srcB) ]
  -> name rewriting due to DNS: map nameB to B's IP address
config[ A(srcA,nameB,dstB,item), B(srcB) ]
  -> name rewriting due to route table lookup: map dstB to gwX
  -> name rewriting due to ARP: map gwX IP name to eth name
config[ A(srcA,nameB,item), lan1(ethX,srcA,dstB,item), B(srcB) ]
  -> delivery at gateway X
config[ A(srcA,nameB,item), X(pkt(srcA,dstB,item)), B(srcB) ]
  -> name rewriting due to route table lookup: map dstB to dstB
  -> name rewriting due to ARP: map dstB IP name to eth name
config[ A(srcA,nameB,item), lan2(ethB,srcA,dstB,item), B(srcB) ]
  -> delivery at B
config[ A(srcA,nameB,item), B(srcB, pkt(srcA,dstB,item)) ]
  -> delivery at application level:
config[ A(srcA,nameB,item), B(srcB,item) ]        # voila: the item was replicated through DNR
```

Name-ReWriting Service, the API

Name-Rewriting as an Abstract Data Type (ADT), basically a key-value store

```
class NaRW: # a name rewriting service, its interface

    def get():     # also called "lookup", "resolve", "compute"

    def put():     # also called "update", "define", "undefine"

    def items():   # also called "walk", "listdir"
```

DNS, BTC, etc are then subclasses, type refinements, interface implementors.

Goal of this “ADT talk” is to abstract away from the implementation details,
define the ADT by its properties, not the implementation

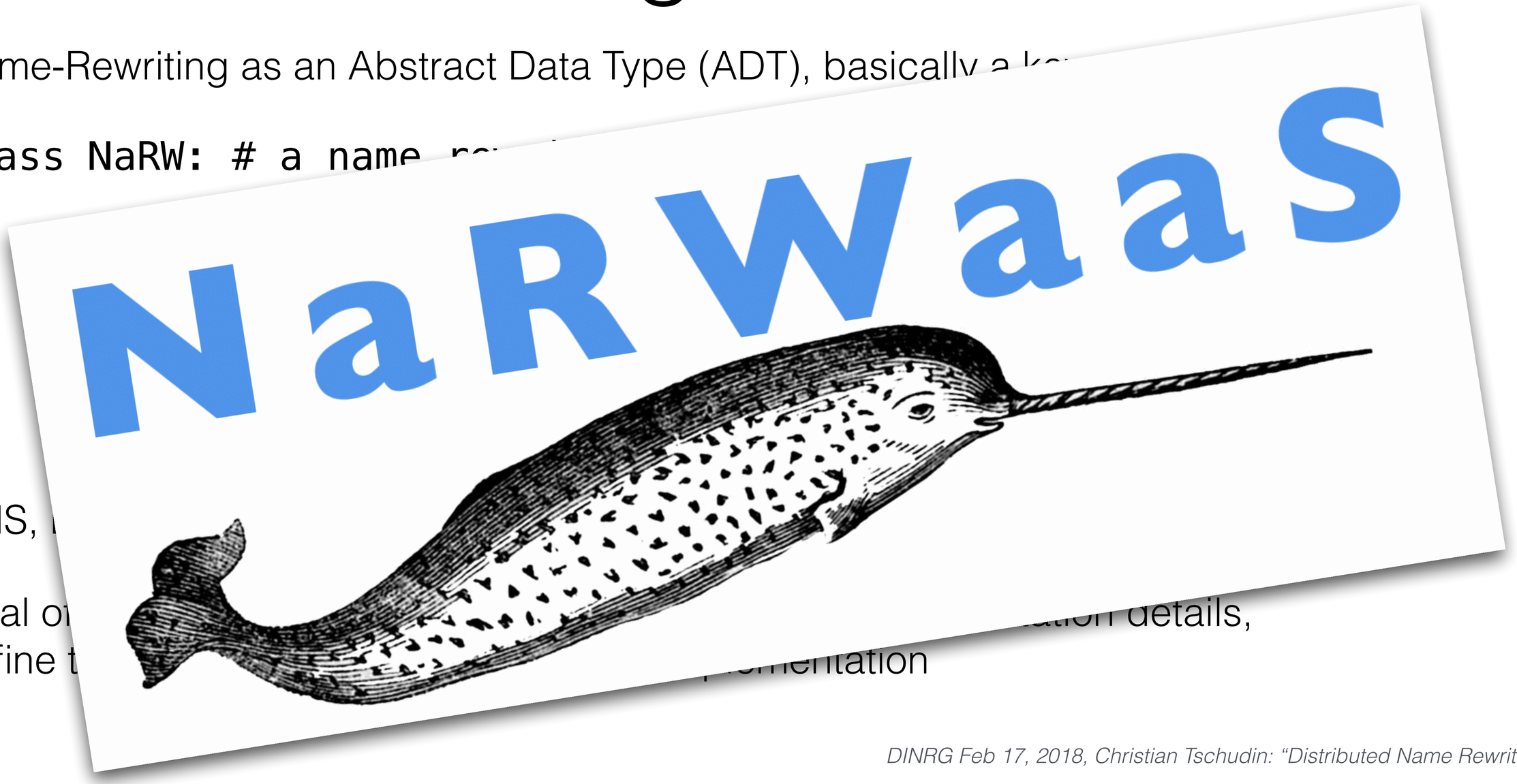
Name-ReWriting Service, the API

Name-Rewriting as an Abstract Data Type (ADT), basically a ke

```
class NaRW: # a name re
```

DNS, I

Goal of
define t



ation details,
mentation

Name-ReWriting Service, the API

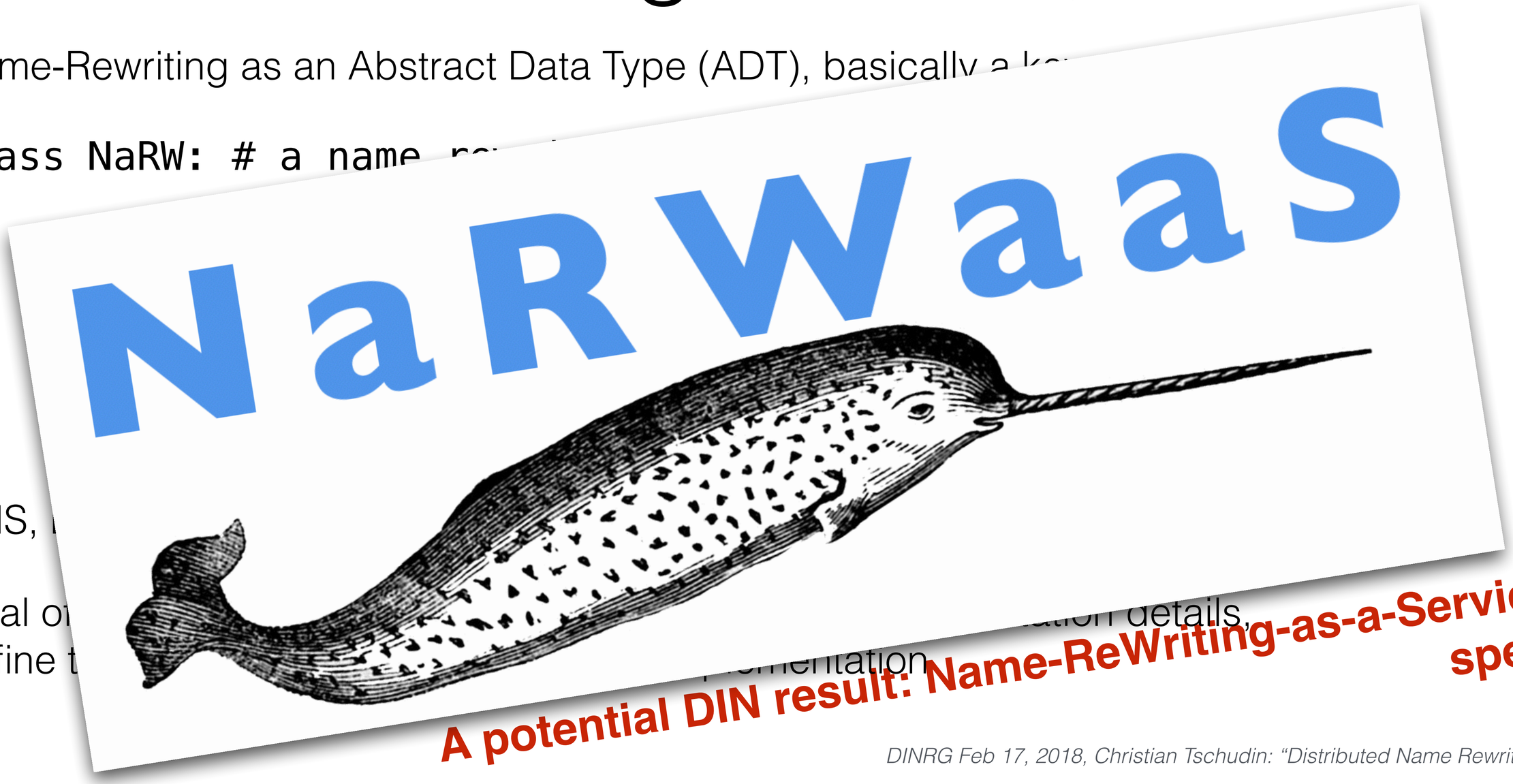
Name-Rewriting as an Abstract Data Type (ADT), basically a ke

```
class NaRW: # a name re
```

DNS, I

Goal of
define t

ation details,
mentation



A potential DIN result: Name-ReWriting-as-a-Service spec

Implementing NaRW with two sub-services

Hypothesis: DIN will revisit these two services over and over

a) **Persistent storage** to store a new item (`lookup(id) -> data`)
take some CRUD database (create,read,update,delete), potentially append-only

b) **head- (or “tip”) service** -- points to the most recent versions of an item

The rest is chaining items to other items via hash pointers (= items’ intrinsic names)

Intuition:

- GitHub, BlockChain (fuses a and b), IOTA’s tangle has multiple tips

- DNS has/is only head-service, ICN offers only storage …

The sweet spot for scalability and trustlessness ?

NaRW API (put,get,items)

“Conflict-Free Replicated Data Types” (CRDT): deterministic eventual **consistency** without consensus, hence **scalable**

—— medium guarantees —- *—— strong guarantees -—*

DNS scales but:
no history, trust-based,
no auto-conflict resolution

persistent storage service

head- (“tip”) service

BlockChain à la **BTC**:
trustless, history, does not scale

tangle-style, w/o consensus

The sweet spot for scalability and trustlessness ?

`NaRW API (put,get,items)`

“Conflict-Free Replicated Data Types” (CRDT): deterministic eventual **consistency** without consensus, hence **scalable**

e.g. scalable key-value store with “observed removal semantics”, or a voting DIN, etc

—— medium guarantees —-

—— strong guarantees -—

DNS scales but:
no history, trust-based,
no auto-conflict resolution

persistent storage service

head-(“tip”) service

BlockChain à la **BTC**:
trustless, history, does not scale

tangle-style, w/o consensus

Questions