

CRDTs for fun and profit

Niklas Gustavsson
ngn@spotify.com
@protocol7



Why?

Distributed rate limiting

Token buckets

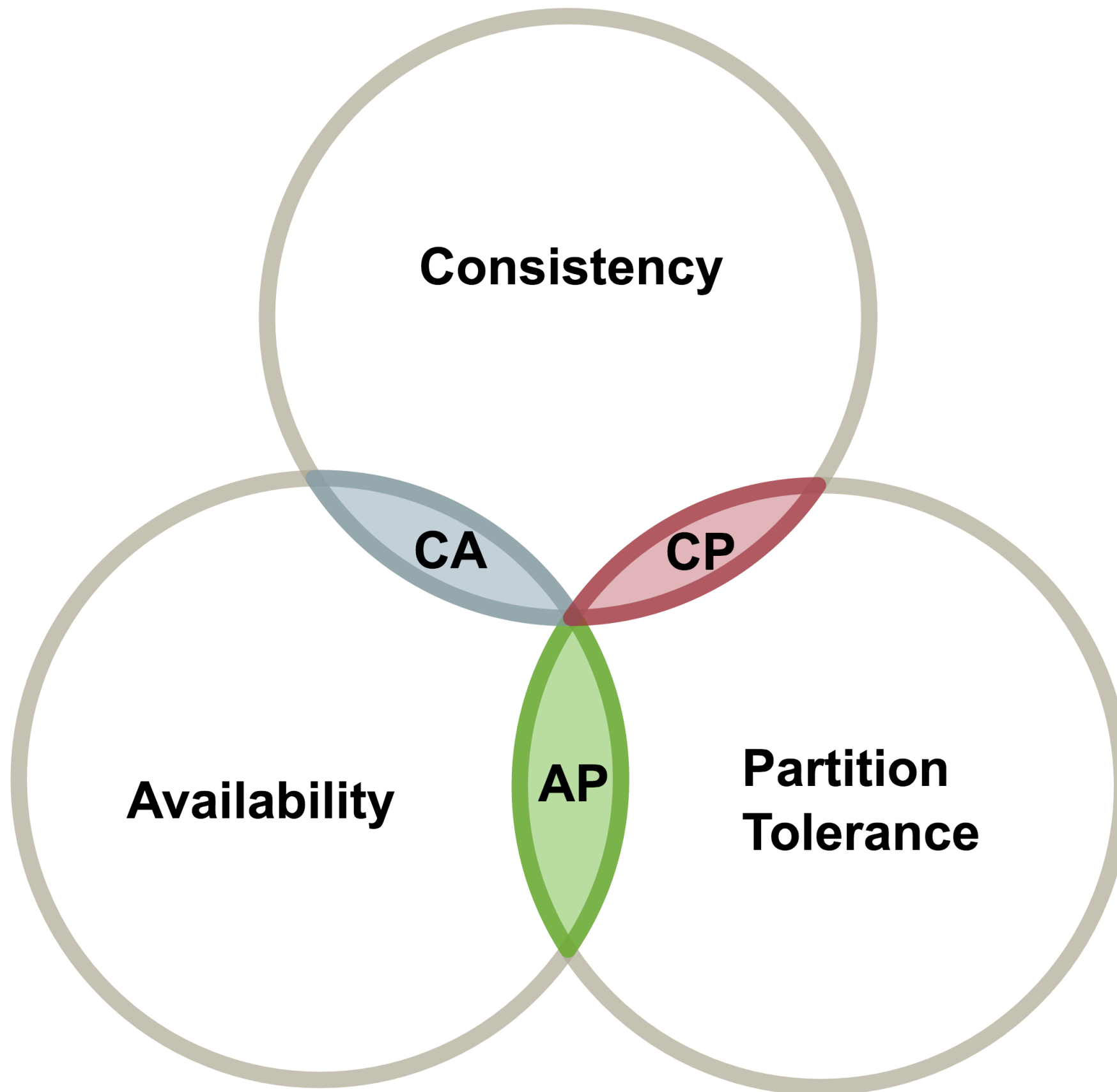
Requirements

Local reads/updates only

Network splits

Best effort

Eventual consistency



CRDTs



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

***A comprehensive study of
Convergent and Commutative Replicated Data Types***

Marc Shapiro, INRIA & LIP6, Paris, France

Nuno Preguiça, CITI, Universidade Nova de Lisboa, Portugal

Carlos Baquero, Universidade do Minho, Portugal

Marek Zawirski, INRIA & UPMC, Paris, France

Conflict-free Replicated Data Type

Convergent Replicated Data Type

Commutative Replicated Data Type

Ops vs state

State based interface

```
interface CRDT<T> {  
    T get();  
  
    void update(T t);  
  
    void merge(CRDT<T> other);  
}
```

Merging

Commutative

$$4+3 = 3+4$$

Associative

$$2 + (3 + 4) = (2 + 3) + 4$$

Idempotence

$$\text{fn}(x) = \text{fn}(\text{fn}(x))$$

Some known CRDTs

Counters

- G counter
- PN counter

- Node 1

```
{  
  node1: 0  
}
```

- Node 2

```
{  
  node2: 0  
}
```

- Node 1

```
{  
  node1: 1  
}
```

- Node 2

```
{  
  node2: 0  
}
```

- Node 1

```
{  
  node1: 1  
}
```

- Node 2

```
{  
  node2: 2  
}
```

- Node 1

```
{  
  node1: 1,  
  node2: 2  
}
```

- Node 2

```
{  
  node1: 1,  
  node2: 2  
}
```

Unbounded growth/GC

Sets/registers

- G set
- 2P set
- LWW set
- LWW register
- Multi value register

- Node 1

```
{  
  node1: { adds: #{}, removes: {} }  
}
```

- Node 2

```
{  
  node2: { adds: #{}, removes: {} }  
}
```

- Node 1

```
{  
  node1: { adds: #{x}, removes: #{} }  
}
```

- Node 2

```
{  
  node2: { adds: #{}, removes: #{} }  
}
```

- Node 1

```
{  
  node1: { adds: #{x}, removes: #{} }  
}
```

- Node 2

```
{  
  node2: { adds: #{y}, removes: #{} }  
}
```

- Node 1

```
{  
  node1: { adds: #{x}, removes: #{} }  
}
```

- Node 2

```
{  
  node2: { adds: #{y}, removes: #{x} }  
}
```

- Node 1

```
{  
  node1: { adds: #{x}, removes: #{} },  
  node2: { adds: #{y}, removes: #{x} }  
}
```

- Node 2

```
{  
  node1: { adds: #{x}, removes: #{} },  
  node2: { adds: #{y}, removes: #{x} }  
}
```

Other examples

- Graphs
- Treedocs
- Maps
- ...

Gossip

"More typically gossip protocols are those that specifically run in a regular, periodic, relatively lazy, symmetric and decentralized manner; the high degree of symmetry among nodes is particularly characteristic. [..] Frequently, the most useful gossip protocols turn out to be those with exponentially rapid convergence towards a state that "emerges" with probability 1.0.

-- https://en.wikipedia.org/wiki/Gossip_protocol

Our implementation

- GCounter

```
{  
  random-node-id: 1  
}
```

- LWW counter

```
{  
  value: 1488746782, timestamp: 1488745432  
}
```

Delta based gossip

Recommended reading

- <https://christophermeiklejohn.com/crdt/2014/07/22/readings-in-crdts.html>
- <http://hal.upmc.fr/inria-00555588/document>
- <https://dist-sys-slack.herokuapp.com/#crdts>
- <https://github.com/aphyr/meangirls>
- @xmal
- <http://docs.basho.com/riak/latest/dev/using/data-types/>

Questions?

ngn@spotify.com
@protocol7

