ActivityPub: un protocollo per reti sociali decentralizzate basato sui Linked Data



cristianolongo@opendatahacklab.org

ActivityPub: un protocollo per reti sociali decentralizzate basato sui Linked Data by Cristiano Longo is marked with **CC0 1.0 Universal**.

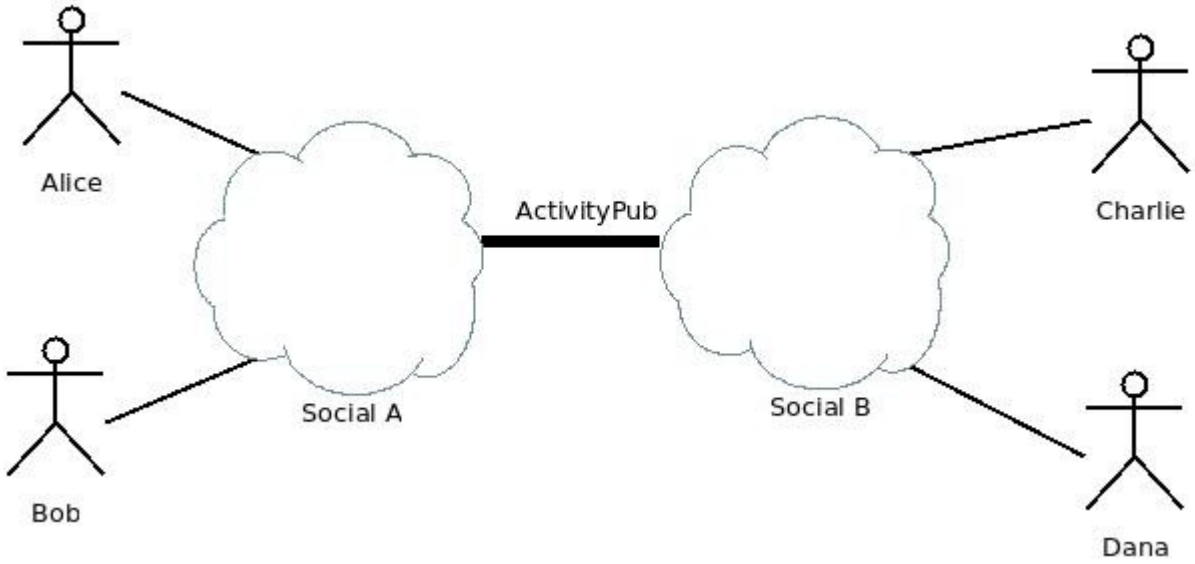
To view a copy of this license, visit https://creativecommons.org/publicdomain/zero/1.0/

Sources

https://www.opendatahacklab.org/ActivityPub/20240612/miniActivityPub.zip

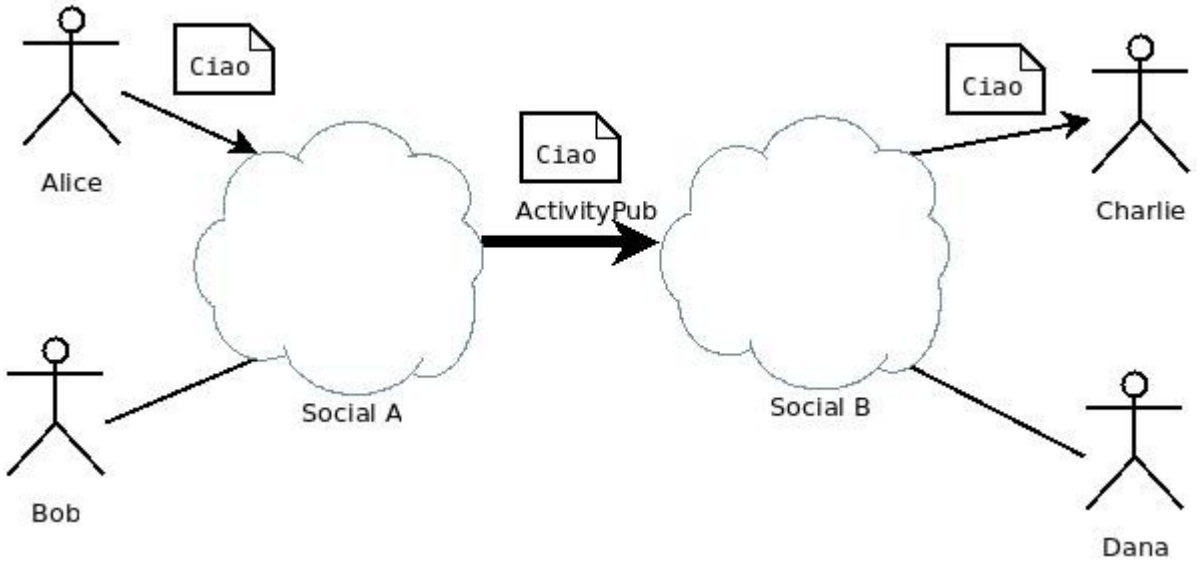
Social Network Decentralizzati

ActivityPub è un protocollo di comunicazione che nasce con lo scopo di permettere ad utenti di social network differenti di comunicare tra loro



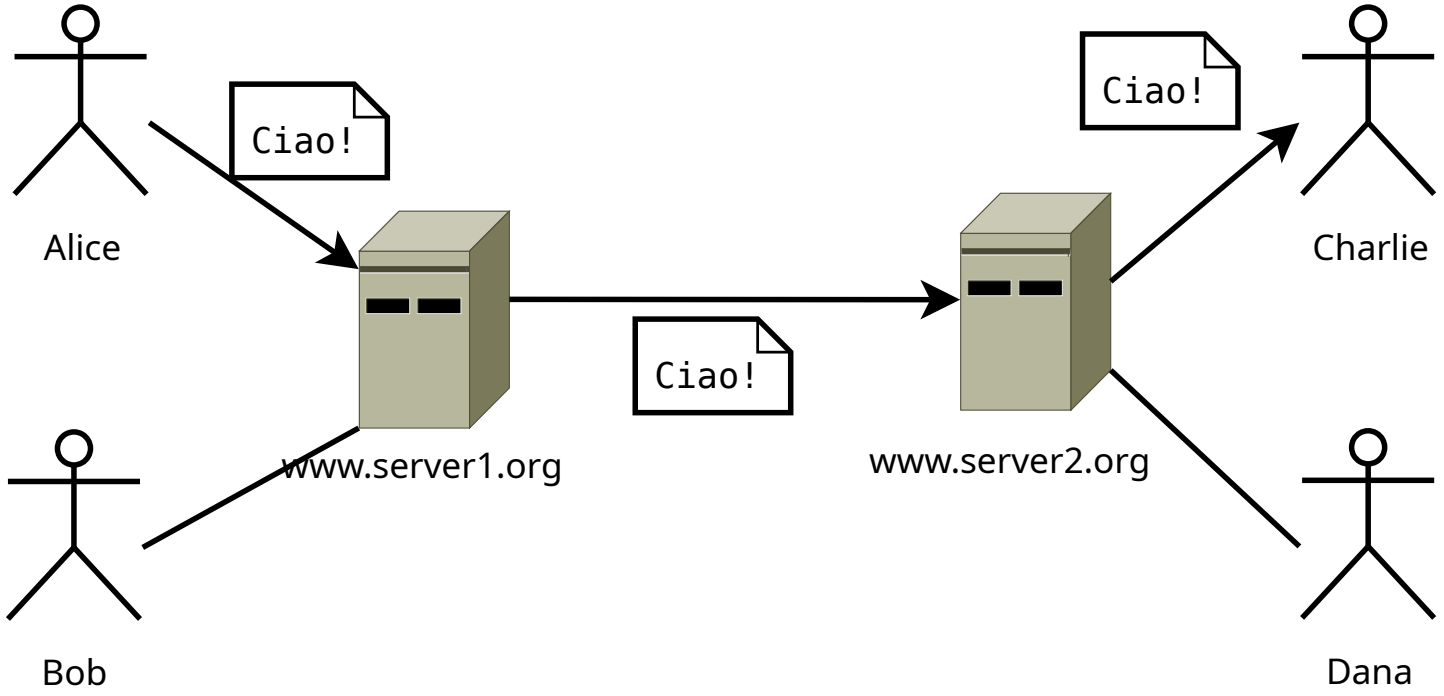
Social Network Decentralizzati

Ad esempio, Alice, che ha un account su *Social A*, può inviare messaggi a Charlie, che ha un account su *Social B.*



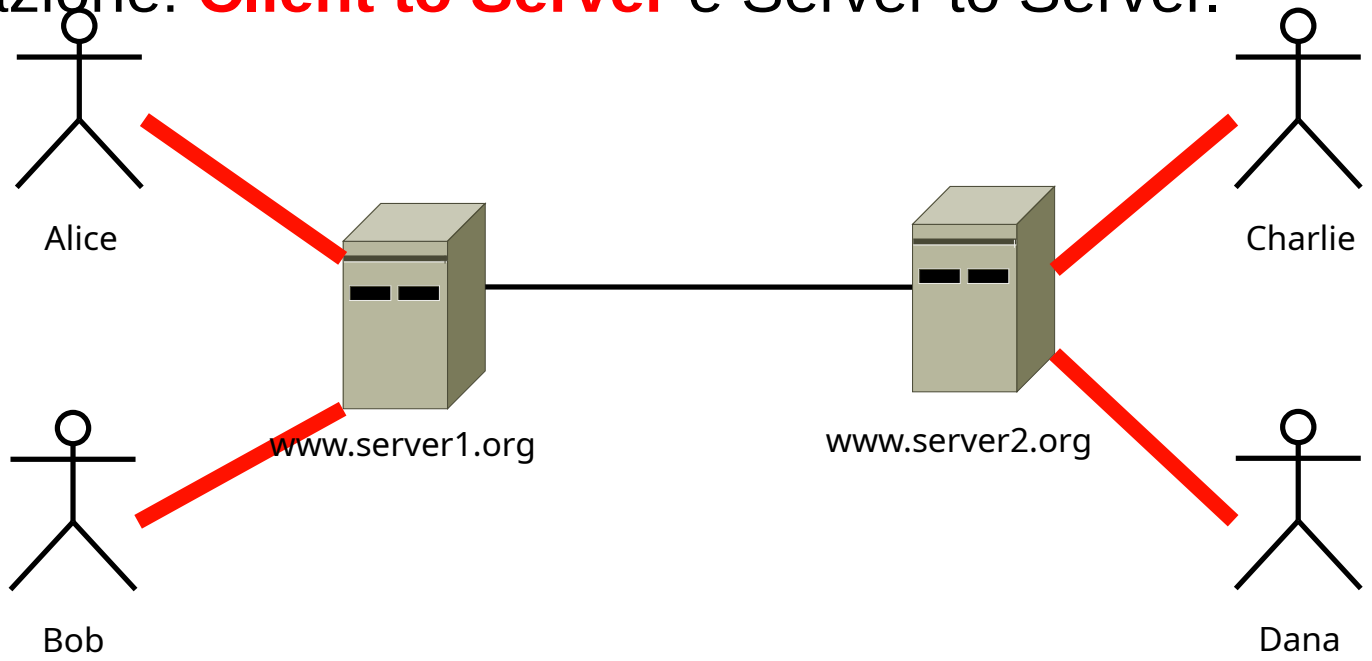
Social Network Decentralizzati

Nel seguito utilizzeremo una semplificazione identificando un social network con un **server**.



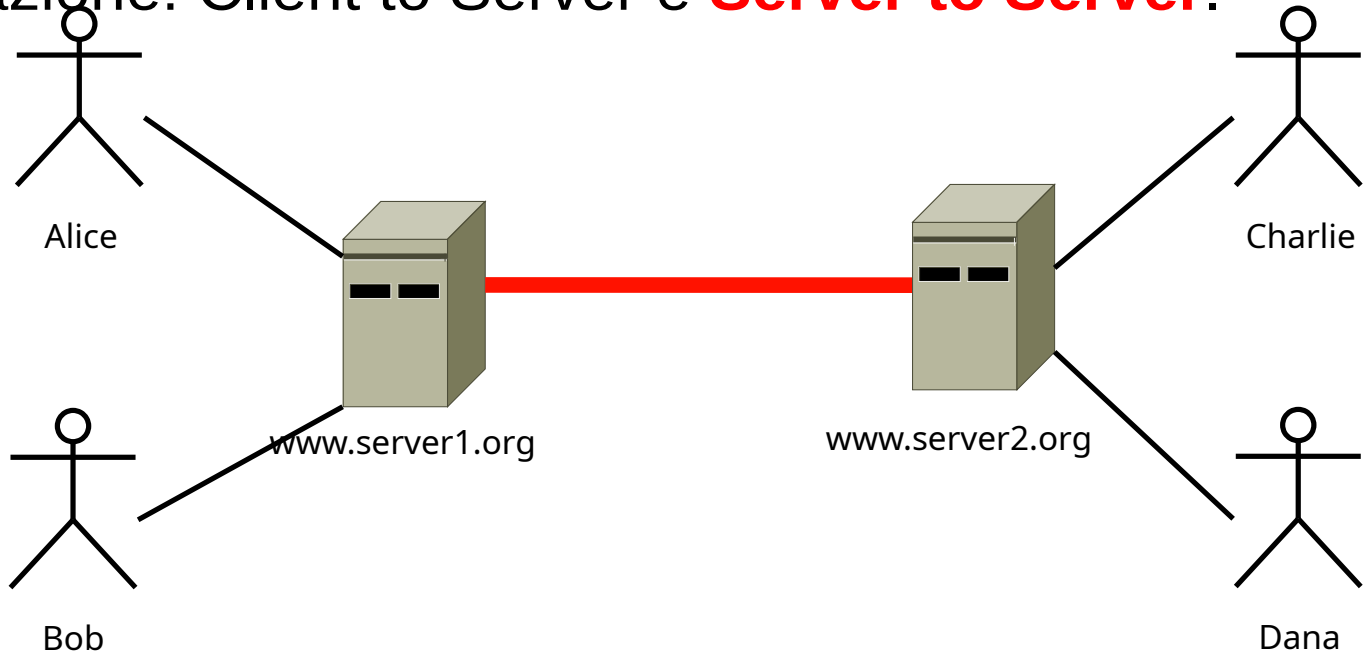
ActivityPub: concetti di base

ActivityPub è una raccomandazione W3C (https://www.w3.org/TR/activitypub/) che definisce da due protocolli di comunicazione: **Client to Server** e Server to Server.



ActivityPub: concetti di base

ActivityPub è una raccomandazione W3C (https://www.w3.org/TR/activitypub/) che definisce da due protocolli di comunicazione: Client to Server e **Server to Server**.

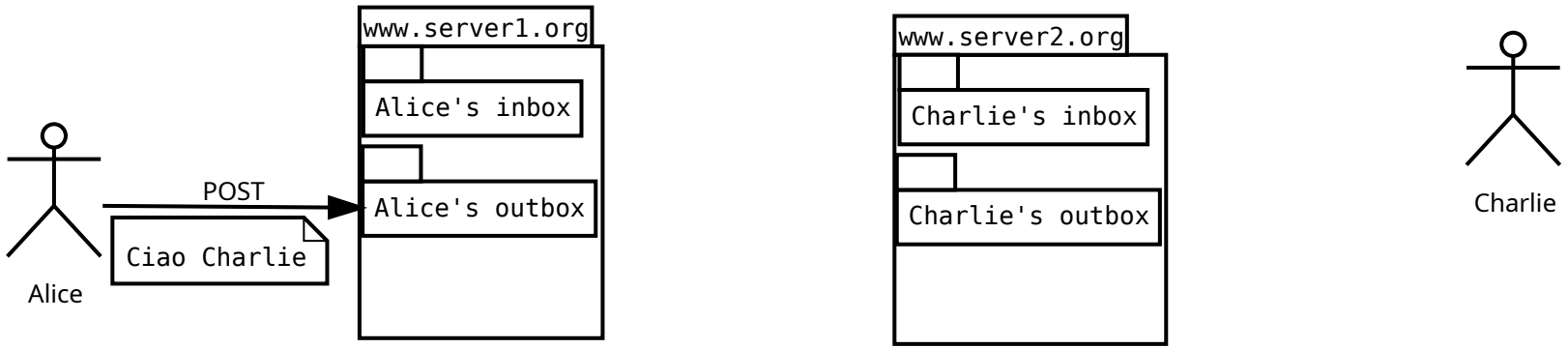


Funzionalità minimali di un server ActivityPub

- pubblicazione del **profilo utente**, contenente informazioni sull’utente e su come contattarlo;
- endpoint **inbox** per ricevere contenuti informativi;
- endpoint **outbox** per inviare contenuti informativi.

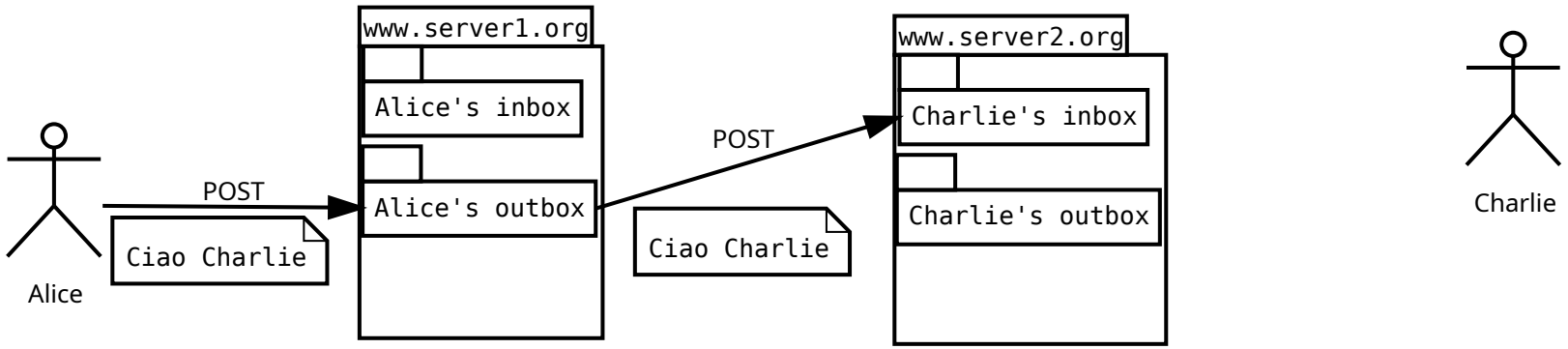
Panoramica del flusso informativo

- Alice inserisce nella propria outbox (HTTP POST) un messaggio specificando Charlie come destinatario;



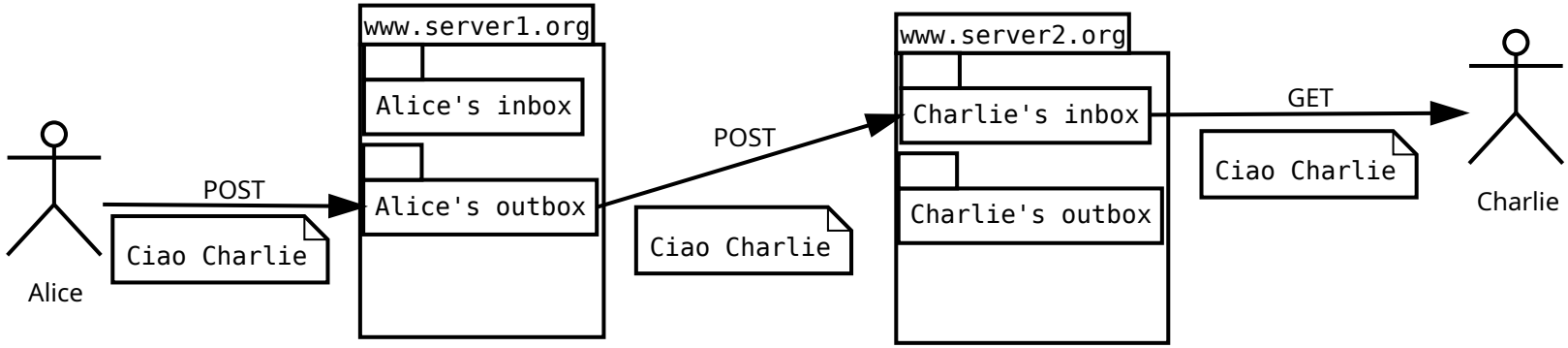
Panoramica del flusso informativo

- Alice inserisce nella propria outbox (HTTP POST) un messaggio specificando Charlie come destinatario;
- il server di Alice scopre l'indirizzo della inbox di Charlie (profilo) e vi inoltra il messaggio (HTTP POST);



Panoramica del flusso informativo

- Alice inserisce nella propria outbox (HTTP POST) un messaggio specificando Charlie come destinatario;
- il server di Alice scopre l'indirizzo della inbox di Charlie (profilo) e vi inoltra il messaggio (HTTP POST);
- Charlie accede alla propria inbox (HTTP GET) e ivi trova il messaggio di Alice.



Common Practices

La raccomandazione ActivityPub non definisce le modalità di l'autenticazione, ma per queste rimanda alle *common practices.* vedi

https://www.w3.org/wiki/SocialCG/ActivityPub/Authentication_Authorization

- Client to Server: OAuth 2.0 (non lo vedremo qui)
- Server to Server: HTTP Signatures (ad ogni utente è associata una **coppia di chiavi RSA)**

Common Practices

La raccomandazione ActivityPub non definisce le modalità di l'autenticazione, ma per queste rimanda alle *common practices.* vedi

https://www.w3.org/wiki/SocialCG/ActivityPub/Authentication_Authorization

- Client to Server: OAuth 2.0 (non lo vedremo qui)
- Server to Server: HTTP Signatures (ad ogni utente è associata una **coppia di chiavi RSA)**

Un altro protocollo largamente usato per il *discovery* nel fediverso è **WebFinger**.

Common Practices : notazione

I protocolli che saranno trattati in seguito e che non rientrano nello standard ActivityPub saranno evidenziati con una sottolineatura.

ActivityStream

Il formato dei messaggi scambiati con ActivityPub è basato su **ActivityStreams** (https://www.w3.org/TR/activitystreams-core/)

ActivityStreams a sua volta è basato su **JSON-LD**

(https://json-ld.org/).

JSON-LD è una serializzazione per **Linked Data** (https://www.w3.org/wiki/LinkedData).

Argomenti

- Linked Data
- JSON-LD
- WebFinger
- Activity Streams + ActivityPub
- HTTP Signatures

Strumenti

Mastodon è l'implementazione più diffusa di ActivityPub. Si vedano

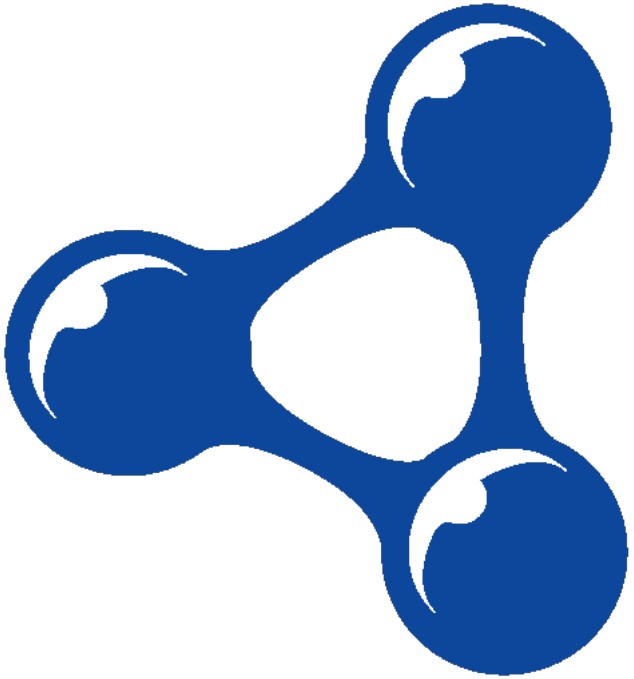
- https://joinmastodon.org/,
- https://docs.joinmastodon.org/spec/activitypub .

LittleActivityPub (https://www.opendatahacklab.org/lap_src/) è un server realizzato a fini didattici che permette l'invio e la ricezione di *Activity* col protocollo ActivityPub.

Per altre implementazioni di ActivityPub si veda

https://joinfediverse.wiki

Linked Data



Linked Data Principles

Da https://www.w3.org/wiki/LinkedData

1) Use URIs as names for things
2) Use HTTP URIs so that people can look up those names
3) When someone looks up a URI, provide useful information
4) Include links to other URIs. so that they can discover more things.

Internationalized Resource Identifiers (IRIs)

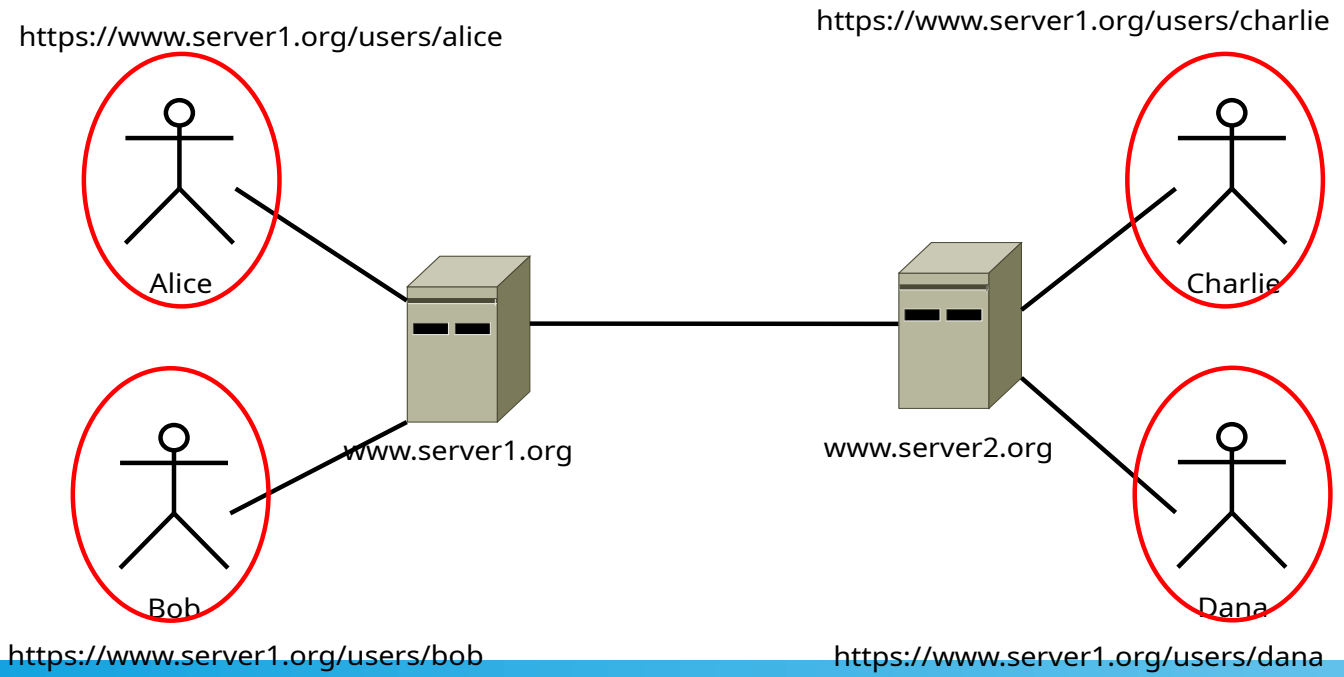
Internationalized Resource Identifiers (IRIs, RFC 3987) estende URI (RFC 3986) permettendo l’utilizzo di caratteri al di fuori del charset US-ASCII.

Nel seguito quando troveremo URI e URL, vanno intese come IRI.

Linked Data Principle 1

1) Use URIs as names for things

Esempio: Gli utenti di un server ActivityPub sono chiamati ***Attori***, **e sono** identificati univocamente da una IRI in un namespace del server.



Linked Data Principle 2

2) Use HTTP URIs so that people can look up those names

https://mastodon.bida.im/users/aaronwinstonsmith restituisce la pagina con le informazioni dell'utente *aaronwinstonsmith* sul server mastodon.bida.im.



Linked Data Principle 3

3) When someone looks up a URI, provide useful information

Richiedendo il media type

- application/ld+json; profile="https://www.w3.org/ns/activitystreams” o
- *application/activity+json*

si ottiene la descrizione dell’utente nel formato *activity streams*.

```
> curl -H ‘Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams”’
https://mastodon.bida.im/users/aaronwinstonsmith
```

Linked Data Principle 4

4) Include links to other URIs. so that they can discover more things

Nel file JSON-LD che rappresenta l'attore è possibile estrapolare, tra le altre cose, la sua inbox.

```
{
    "@context": [
        "https://www.w3.org/ns/activitystreams","https://w3id.org/security/v1", { .. }],
    "id": "https://mastodon.bida.im/users/aaronwinstonsmith",
    "type": "Person",
    "inbox": "https://mastodon.bida.im/users/aaronwinstonsmith/inbox",
    "outbox": "https://mastodon.bida.im/users/aaronwinstonsmith/outbox",
    …
}
```

Semantic Web Stack

I Linked Data devono essere forniti usando I linguaggi del **Web Semantico** (https://www.w3.org/2001/sw/):

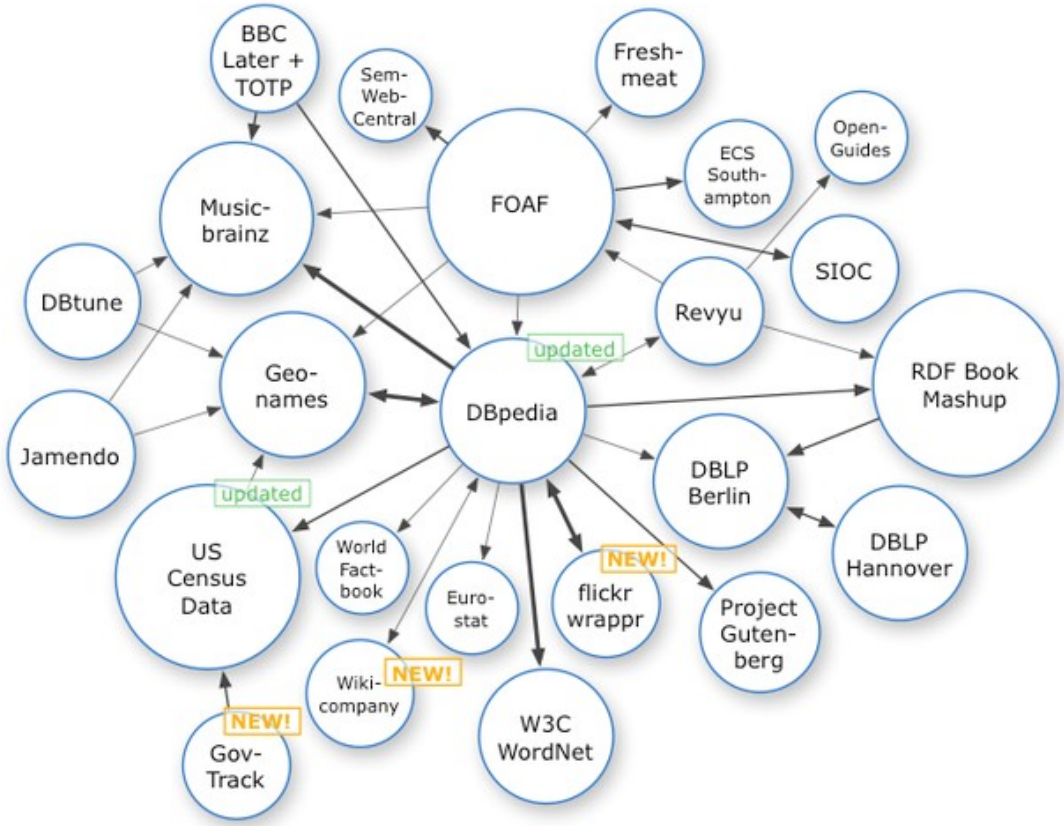
RDF, RDFS, OWL2, SPARQL.



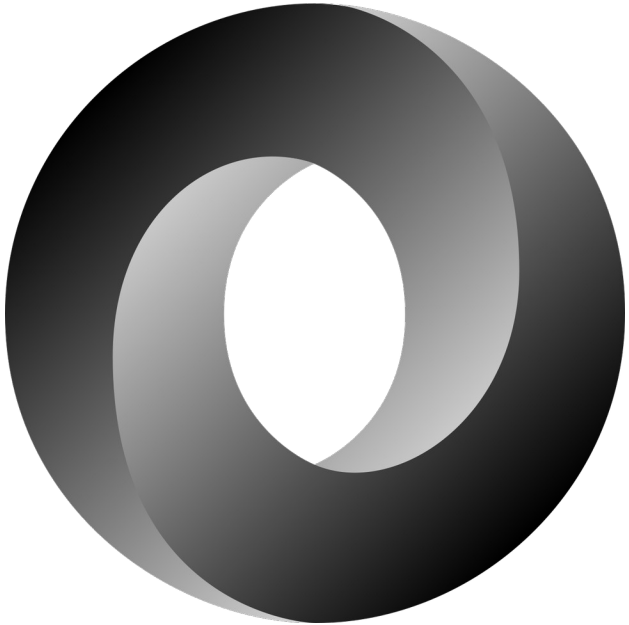
Linked Open Data Cloud

I dataset pubblicati come Linked Data possono contenere **collegamenti** ad altri dataset

(vedi https://lod-cloud.net/)



JSON



JSON

Javascript Object Notation (JSON, (RFC 8259) è un formato *testuale* per lo scambio di dati.

JSON - Object

Un **oggetto** JSON è un insieme finito di coppie

“*attributo*” : *valore*

```
  "nome": "Cristiano"
```

JSON - Object

Un **oggetto** JSON è un insieme finito di coppie

“*attributo*” : *valore*

separate da “,”

```
  "nome": "Cristiano",
  "cognome":  "Longo"
```

JSON - Object

Un **oggetto** JSON è un insieme finito di coppie

“*attributo*” : *valore*

separate da “,” e racchiuso tra “{“ e “}”

```
{
  "nome": "Cristiano",
  "cognome":  "Longo"
}
```

JSON - valori

Come valore è possibile specificare

- una stringa

```
{
  "nome": "Cristiano",
  "cognome":  "Longo"
}
```

JSON - valori

Come valore è possibile specificare

- una stringa,
- un numero

```
{
  "nome": "Cristiano",
  "cognome":  "Longo",
  "anni": 18
}
```

JSON - valori

Come valore è possibile specificare

- una stringa,
- un numero,
- true o false

```
{
  "nome": "Cristiano",
  "cognome":  "Longo",
  "anni": 18,
  "ancoraVivo": true
}
```

JSON - valori

Come valore è possibile specificare

- una stringa,
- un numero,
- true o false,
- un altro oggetto,
- un *array.*

```
{
  "nome": "Cristiano",
  "cognome":  "Longo",
  "anni": 18,
  "ancoraVivo": true,
  "indirizzo":{
    "via": "Via dei Matti",
    "numero": 0
  },
  "cittadinanza":["italiana",
                  "neozelandese]
}
```

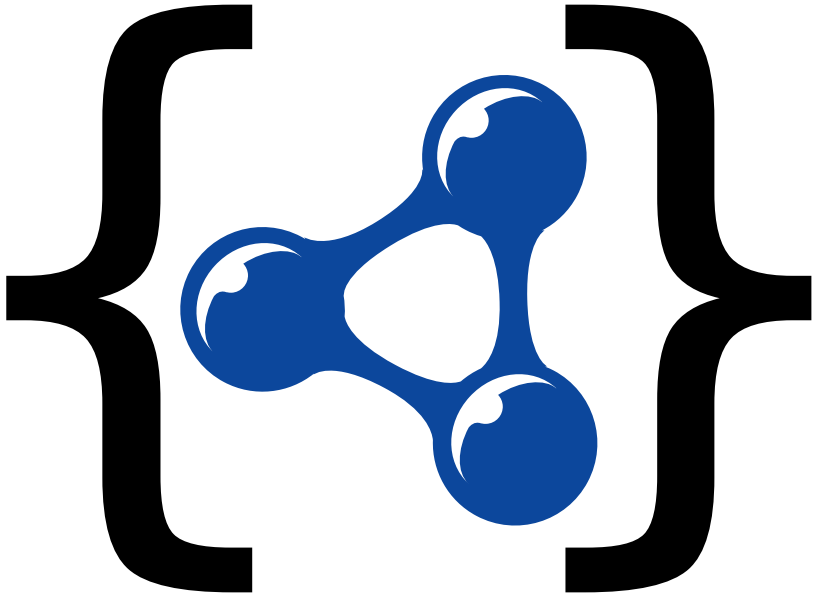
JSON - valori

Come valore è possibile specificare

- una stringa,
- un numero,
- true o false,
- un altro oggetto,
- un **array*:*** lista *di valori separati da “,” e racchiusa tra “[“ e “]”.*

```
{
  "nome": "Cristiano",
  "cognome":  "Longo",
  "anni": 18,
  "ancoraVivo": true,
  "indirizzo":{
    "via": "Via dei Matti",
    "numero": 0
  },
  "cittadinanza":["italiana",
                  "neozelandese]
}
```

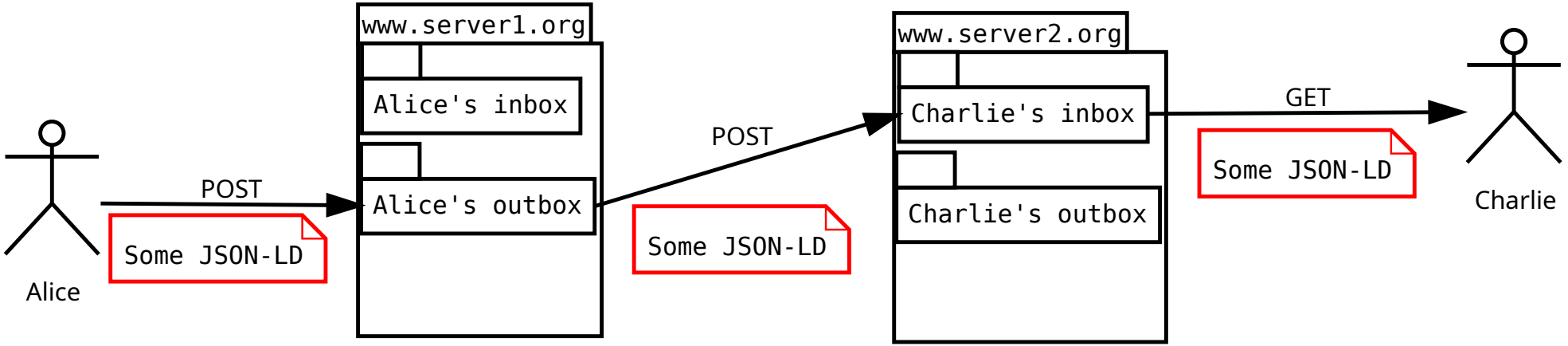
JSON-LD



JSON-LD

Il *payload* dei messaggi è in formato JSON-LD.

JSON-LD (https://json-ld.org/,https://www.w3.org/TR/json-ld/) è una serializzazione di RDF basata su JSON (RFC8259).



JSON-LD Keywords

Le *keyword* di JSON-LD sono stringhe alfanumeriche *case-sensitive* definite in https://www.w3.org/TR/json-ld/

Tutte le keyword di JSON-LD iniziano col carattere @

Node objects

Un *node object* è un oggetto (nel senso di JSON) che rappresenta una risorsa RDF.

Con la keyword *@id* è possibile specificare la IRI della risorsa (o l'identificativo in caso di blank node)

JSON-LD

```
{
  "@id": "http://test.org/i",
  "http://test.org/p": "data"
}
```

RDF

```
<http://test.org/i> <http://test.org/p> "data"
```

Proprietà

Gli attributi che non cominciano per “@” sono le proprietà dell’oggetto

JSON-LD

```
{
  "@id": "http://test.org/i",
  "http://test.org/p": "data"
}
```

RDF

```
<http://test.org/i> <http://test.org/p> "data"
```

Proprietà

Gli attributi che non cominciano per “@” sono le proprietà dell’oggetto.

Per specificare valori multipli per una proprietà si usano gli array.

JSON-LD

```
{
  "@id": "http://test.org/i",
  "http://test.org/p": ["data1", "data2"]
}
```

RDF

```
<http://test.org/i> <http://test.org/p> "data1" .
<http://test.org/i> <http://test.org/p> "data2" .
```

Value objects

I *value object* si usano per rappresentare dei *letterali* (nel senso di RDF).

Un *value object* è un oggetto JSON con l'attributo *@value.*

JSON-LD

```
{
  "@id": "http://www.example.org/somedocument",
  "http://www.example.org/modified" :{
    "@value": "2010-05-29T14:17:39+02:00",
    "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
  }
}
```

RDF

<http://www.example.org/somedocument> <http://www.example.org/modified> "2010-05-29T14:17:39+02:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> .

List objects

I list *object* sono oggetti JSON con l'attributo @list che definisce una lista ordinata di valori o oggetti.

JSON-LD

```
{
  "@id": "http://test.org/i",
  "http://test.org/p":{
    "@list": ["data1", "data2"]
  }
}
```

RDF

```
<http://test.org/i> <http://test.org/p> _:b0 .
_:b0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> "data1" .
_:b0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> _:b1 .
_:b1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> "data2" .
_:b1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> <http://www.w3.org/1999/02/22-rdf-syntax-ns#nil> .
```

Classi in JSON-LD

@type è una keyword per indicare la **classe** cui appartiene la risorsa

JSON-LD

```
{
  "@id": "http://test.org/i",
  "@type": "http://test.org/c"
}
```

RDF

```
<http://test.org/i> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://test.org/c>
```

Graph objects

Un *Graph object* rappresenta un grafo RDF. È un documento JSON con un attributo *@graph* che ha come valore un array di node object che descrivono le triple del grafo.

JSON-LD

```
{
	"@graph": [
		{
			"@id": "http://test.org/i",
			"@type": "http://test.org/c"
		}
	]
}
```

Named graphs

Ad un *Graph object* può essere associata una IRI, sempre mediante l'attributo @id, ed eventualmente altri metadati.

JSON-LD

```
{
      "@id": "http://test.org/mygraph"
      "@graph": [
            {
                  "@id": "http://test.org/i",
                  "@type": "http://test.org/c"
            }
      ]
}
```

@context

Mediante l'attributo @context in un oggetto JSON-LD è possibile definire alcune regole di espansione per i termini presenti nell'oggetto.

JSON-LD

```
{
      "@context": {
            "p": "http://test.org/p"
      },
      "@id": "http://test.org/i",
      "p": ["data1", "data2"]
}
```

RDF

```
<http://test.org/i> <http://test.org/p> "data1" .
<http://test.org/i> <http://test.org/p> "data2" .
```

@context/@vocab

Mediante l’attributo *@vocab* all’interno di un context si definisce il namespace per classi e proprietà.

JSON-LD

```
{
      "@context": {
            "@vocab": "http://test.org/vocab#"
      },
      "@id": "http://test.org/i",
      "p": ["data1", "data2"]
}
```

RDF

```
<http://test.org/i> <http://test.org/vocab#p> "data1" .
<http://test.org/i> <http://test.org//vocab#p> "data2" .
```

@context/@base

Con l’attributo *@base* all’interno di un context si definisce il namespace contro il quale verranno risolte le IRI degli oggetti.

JSON-LD

```
{
      "@context": {
            "@base": "http://test.org/",
            "@vocab": "http://test.org/vocab#",
      },
      "@id": "i",
      "p": ["data1", "data2"]
}
```

RDF

```
<http://test.org/i> <http://test.org/vocab#p> "data1" .
<http://test.org/i> <http://test.org//vocab#p> "data2" .
```

Prefissi

Nel context è possibile definire dei *prefissi* che verranno utilizzati per risolvere termini del tipo <*prefisso*>:<*nome*>

JSON-LD

```
{
      "@context": {
            "foaf": "http://xmlns.com/foaf/0.1/"
      },
   "@id": "http://test.org/i",
   "foaf:name": "Cristiano Longo"
}
```

RDF

```
<http://test.org/i> <http://xmlns.com/foaf/0.1/name> "Cristiano Longo" .
```

Alias per le keyword

Nel context è possibile definire delle *alias* per tutte le keyword di JSON-LD, ad eccezione di @context

JSON-LD

```
{
      "@context": {
            "foaf": "http://xmlns.com/foaf/0.1/",
            "id": "@id"
      },
      "id": "http://test.org/i",
      "foaf:name": "Cristiano Longo"
}
```

RDF

```
<http://test.org/i> <http://xmlns.com/foaf/0.1/name> "Cristiano Longo" .
```

Type Coercion

Sempre nel context è possibile specificare per ogni proprietà il *tipo* che sarà assegnato agli oggetti (target) associati alla stessa.

JSON-LD

```
{
	"@context": {
		"modified": {
			"@id": "http://www.example.org/modified",
			"@type": "http://www.w3.org/2001/XMLSchema#dateTime"
		}
	},
	"@id": "http://www.example.org/somedocument",
	"modified": "2010-05-29T14:17:39+02:00"
}
```

RDF

```
<http://www.example.org/somedocument> <http://www.example.org/modified> "2010-05-29T14:17:39+02:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
```

Object properties

Nel caso in cui il valore della proprietà siano risorse RDF, è possibile specificare nel context “@id” come “@type”

JSON-LD

```
{
        "@context": {
                "homepage": {
                        "@id": "http://xmlns.com/foaf/0.1/homepage",
                        "@type": "@id"
        }
    },
        "@id": "http://www.example.org/me",
        "homepage": "http://myhome.org"
}
```

RDF

```
<http://www.example.org/me> <http://xmlns.com/foaf/0.1/homepage> <http://myhome.org> .
```

Definizione di liste

Sempre nel context è anche possibile forzare il valore di una proprietà a lista specificando “@container”: “@list”

JSON-LD

```
{
  "@context": {
    "p": {
      "@id": "http://test.org/p",
      "@container": "@list"
    }
  },
  "@id": "http://test.org/i",
  "p": ["data1", "data2"]
}
```

RDF

```
<http://test.org/i> <http://test.org/p> _:b0 .
_:b0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> "data1" .
_:b0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> _:b1 .
_:b1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> "data2" .
_:b1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> <http://www.w3.org/1999/02/22-rdf-syntax-ns#nil> .
```

Language maps

Una datatype property può essere definita come *language map*, che assume come valore una mappa <lingua>: “<testo>”, ove la lingua è un language tag definito secondo https://tools.ietf.org/html/bcp47

JSON-LD

```
{
  "@context": {
    "description": { "@id": "http://purl.org/dc/elements/1.1/description", "@container": "@language"}
  },
  "@id": "http://romans.org/juliuscaesar",
  "description": {
    "it": "Imperatore romano",
    "en": "Roman emperor"
  }
}
```

RDF

```
<http://romans.org/juliuscaesar> <http://purl.org/dc/elements/1.1/description> "Imperatore romano"@it .
<http://romans.org/juliuscaesar> <http://purl.org/dc/elements/1.1/description> "Roman emperor"@en .
```

@context on-line

Se il valore di @context è una IRI, il context utilizzato sarà quello disponibile all'indirizzo specificato.

{**"@context": "https://www.w3.org/ns/activitystreams"**,
"type": "Person",
"id": "https://social.example/alyssa/",
"name": "Alyssa P. Hacker",
"preferredUsername": "alyssa",
"summary": "Lisp enthusiast hailing from MIT",
"inbox": "https://social.example/alyssa/inbox/",
"outbox": "https://social.example/alyssa/outbox/",
"followers": "https://social.example/alyssa/followers/",
"following": "https://social.example/alyssa/following/",
"liked": "https://social.example/alyssa/liked/"}

@context multivalore

È possibile specificare molteplici valori di @context

```
{
   "@context": [
      "https://www.w3.org/ns/activitystreams",
      "https://w3id.org/security/v1",
      {
         "manuallyApprovesFollowers": "as:manuallyApprovesFollowers",
         "toot": "http://joinmastodon.org/ns#",
         "featured": {
            "@id": "toot:featured",
            "@type": "@id"
         },
...
```

Compaction Algorithm

https://www.w3.org/TR/json-ld11-api/#compaction-algorithm

This algorithm compacts a JSON-LD document, such that the given context is applied. This must result in shortening

- *any applicable IRIs to terms or compact IRIs,*
- *any applicable keywords to keyword aliases, and*
- *any applicable JSON-LD values expressed in expanded form to simple values such as strings or numbers.*

WebFinger



WebFinger

In molte altre applicazioni del fediverso, un utente è identificato con

@<username>@<serverdomain>

Ad esempio @aaronwinstonsmith@mastodon.bida.im.

WebFinger

In molte altre applicazioni del fediverso, un utente è identificato con

@<username>@<serverdomain>

Ad esempio @aaronwinstonsmith@mastodon.bida.im.

Come risalire dallo username alla IRI che identifica l’attore? Usando il protocollo *WebFinger* (RFC 7033)

WebFinger Query

Una query WebFinger è una richiesta GET che ha la forma

`https://<server>/.well-known/webfinger?resource=<resource>`

Dove `<server>` è l'indirizzo del server cui inviare la richiesta e `<resource>` è la IRI della risorsa della quale si richiedono informazioni.

Ad esempio

```
https://mastodon.bida.im/.well-known/webfinger?
resource=acct:aaronwinstonsmith@mastodon.bida.im
```

WebFinger Reply

In risposta si ottiene un documento in formato JSON Resource Descriptor (JRD, RFC 6415)

Questo contiene, tra le altre cose, un link con

- *rel=self* e
- *type=application/activity+json*

che punta alla URL dell’attore.

Activity Streams + Activity Pub Data Format



Activity Vocabulary

- Il formato dati utilizzato da **ActivityPub**

  (https://www.w3.org/TR/activitypub)
- è una estensione di **Activity Streams 2.0**

  (https://www.w3.org/TR/activitystreams-core/)
- che, a sua volta, è una sintassi di serializzazione di **Activity Vocabulary**

  (https://www.w3.org/TR/activitystreams-vocabulary/)

Activity Streams Document

Un **Activity Streams Document** è

- Un JSON-LD object di classe *Object* (come definita nell'Activity Vocabulary),
- codificato in UTF-8,
- con MIME media type *application/activity+json,*
- Nel quale è definito almeno il context pubblicato alla seguente IRI

  https://www.w3.org/ns/activitystreams
- consistente con l'output del Compaction Algorithm.

Activity Streams MIME media type

Un **Activity Streams Document** è

- Un JSON-LD object di classe *Object*, codificato in UTF-8,
- con MIME media type *application/activity+json,*
- ….

ActivityPub contempla anche il media type

- application/ld+json; profile="https://www.w3.org/ns/activitystreams"

Esempio di Activity Streams Document

{**"@context": "https://www.w3.org/ns/activitystreams"**,
 "type": "Person",
 "id": "https://social.example/alyssa/",
 "name": "Alyssa P. Hacker",
 "preferredUsername": "alyssa",
 "summary": "Lisp enthusiast hailing from MIT",
 "inbox": "https://social.example/alyssa/inbox/",
 "outbox": "https://social.example/alyssa/outbox/",
 "followers": "https://social.example/alyssa/followers/",
 "following": "https://social.example/alyssa/following/",
 "liked": "https://social.example/alyssa/liked/"
}

Person è una sottoclasse di *Object*

Activity Streams 2.0 JSON-LD @context

Per ottenere il context definito per Activity Streams 2.0

```
curl -H "Accept: application/ld+json" https://www.w3.org/ns/activitystreams
```

ATTENZIONE: questo context contiene anche i termini definiti nelle estensioni ufficiali di Activity Streams 2.0, tra cui ActivityPub.

Activity Vocabulary

In Activity Vocabulary (https://www.w3.org/TR/activitystreams-vocabulary) sono definite classi e proprietà di Activity Streams 2.0.

Namespace https://www.w3.org/ns/activitystreams#

È disponibile una versione OWL2, non normativa e nella serializzazione text/turtle

https://www.w3.org/ns/activitystreams-owl

Gerarchia delle classi

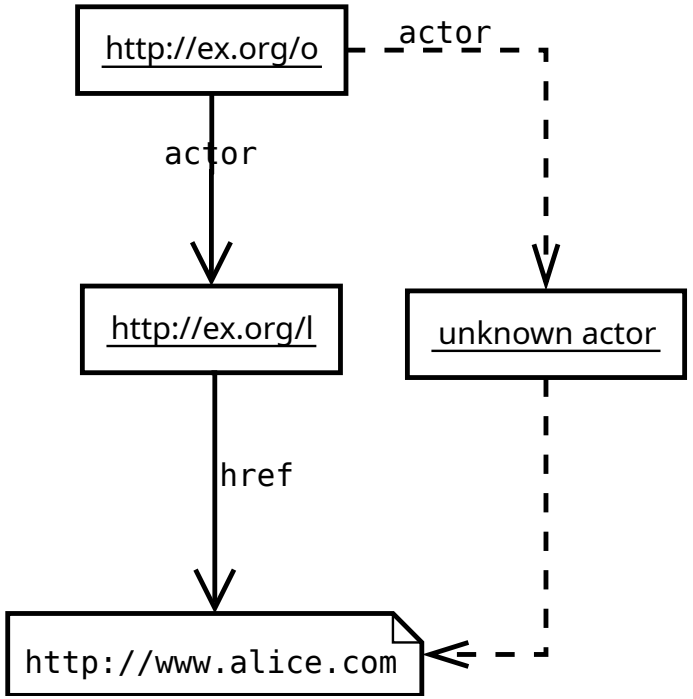
Le classi base di Activity Streams, come definite nell'Activity Vocabulary, sono:

- **Link** permette di referenziare risorse web disponibili in formati e modalità al di fuori di Activity Streams;
- **Object** è la classe base per tutti gli oggetti Activity Streams.

Link

When a Link is used, it establishes a qualified relation connecting the subject (the containing object) to the resource identified by the href.

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "http://example.org/alice",
  "type": "Object",
  "actor": {
    "id": "http://example.org/alicewebsite",
    "type": "Link",
      "href": "http://www.alice.com",
      "hreflang": "en",
      "mediaType": "text/html",
  }
}
```



La classe Object

Gli attori a loro volta possono essere di vari tipi

- Link
- **Object**
  - *Attori*
  - Activity
  - Collection
  - *Contenuti*
  - ...

Attori

Gli attori sono *oggetti capaci di eseguire attività*. In Activity Streams sono definite alcune sottoclassi di Object per rappresentare gli attori.

- Link
- Object
  - **Application**
  - **Group**
  - **Organization**
  - **Person**
  - **Service**
  - *...*

Proprietà degli Attori

Di ogni attore si possono specificare varie proprietà: nome, icona, sommario, ...

```
{
      "@context": [
            "https://www.w3.org/ns/activitystreams","https://w3id.org/security/v1", { .. }],
      "id": "https://mastodon.bida.im/users/aaronwinstonsmith",
      "type": "Person",
      "name": "Aaron Winston Smith",
      "summary": "Ciao sono Winston ... “,

…
}
```

Attori in Activity Pub

In ActivityPub, la **inbox** e la **outbox** sono obbligatorie.

Vedi https://www.w3.org/TR/activitypub/#actor-objects.

```
{
      "@context": [
            "https://www.w3.org/ns/activitystreams","https://w3id.org/security/v1", { ..
}],
      "id": "https://mastodon.bida.im/users/aaronwinstonsmith",
…
       "inbox": "https://mastodon.bida.im/users/aaronwinstonsmith/inbox",
      "outbox": "https://mastodon.bida.im/users/aaronwinstonsmith/outbox",
…
}
```

Altre proprietà per gli attori in Activity Pub

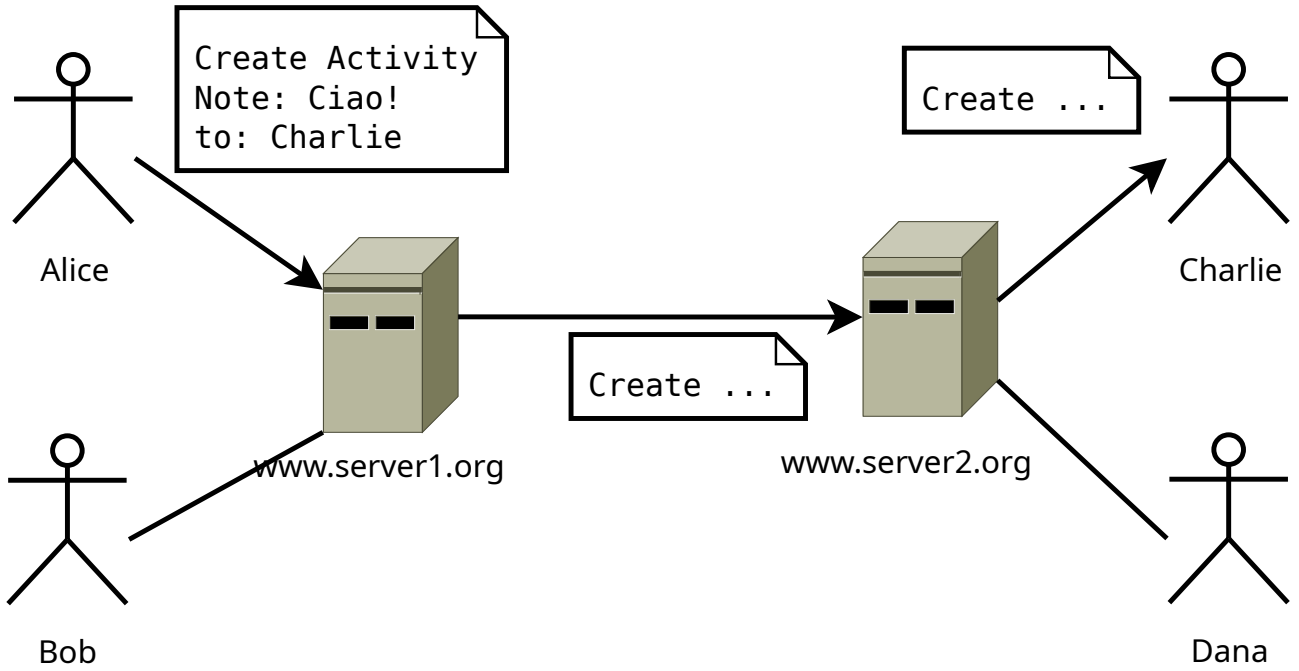
ActivityPub definisce anche altre proprietà applicabili agli attori:

- *preferredUsername* indica il nome utente sul server
- *sharedInbox* è una inbox condivisa da molteplici utenti
- le *Collections followers, followed, likes, liked*, …
- ...

Activity

I messaggi scambiati dagli utenti usando ActivityPub sono istanze della classe **Activity**.

- Object
  - Actor
  - Activity
    - Create
    - Update
    - Delete
    - Follow
    - Accept
    - ...



Proprietà delle Activity

Una Activity rappresenta una azione potrebbe avvenire, è in corso oppure è avvenuta.
https://www.w3.org/TR/activitystreams-vocabulary/#dfn-activity

- con *actor* è indicato l'attore che esegue l'azione
- con *object* è indicato l'oggetto dell'azione


```
{
   "@context": "https://www.w3.org/ns/activitystreams",
   "id": "https://mastodon.bida.im/3bc8a42b-50c0-4fa5-a5d9-e9c273bda6c5",
   "type": "Follow",
   "actor": "https://mastodon.bida.im/users/aaronwinstonsmith",
   "object": "https://www.opendatahacklab.org/social/we.json"
}
```

Activity riguardanti i profili

Follow, Accept, Reject, Block, Flag, Move, …

Al seguente indirizzo l’implementazione di mastodon

https://docs.joinmastodon.org/spec/activitypub/#supported-activities-for-profiles

Esempio – Accettare una follow request

In object specificare la activity ricevuta come follow request.

```
{
    "@context": "https://www.w3.org/ns/activitystreams",
    "id": "https://www.opendatahacklab.org/activities/1234”
    "type": "Accept",
    "actor": "https://www.opendatahacklab.org/social/we.json",
    "object": "https://mastodon.bida.im/3bc8a42b-50c0-4fa5-a5d9-e9c273bda6c5"
}
```

Activity per i contenuti

Create, Update, Delete, Like, Undo, Announce, Flag

Al seguente collegamento è indicato come queste attività vengono trattate da Mastodon.

https://docs.joinmastodon.org/spec/activitypub/#supported-activities-for-statuses

Contenuti

Gli oggetti informativi scambiati mediante activity sono

Note, Question, Article, Page, Image, Audio, Video, Event

Al seguente collegamento sono indicati i contenuti gestiti da mastodon.

https://docs.joinmastodon.org/spec/activitypub/#payloads

Stati

Gli oggetti di tipo *Note* sono assimilabili al concetto di *stato* nel *microblogging*.


```
{
        "id": "https://www.opendatahacklab.org/social/create14/object.json",
        "type": "Note",
        "published": "2023-02-23T21:11:48Z",
        "attributedTo": "https://www.opendatahacklab.org/social/we.json",
        "to": "https://www.w3.org/ns/activitystreams#Public",
        "cc": "https://www.opendatahacklab.org/social/followers.json",
        "sensitive": false,
        "content": "<p>E non ammetto repliche!</p>"
}
```

Esempio: inviare uno stato


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://www.opendatahacklab.org/lap_users/testmaster/activity/newstatus",
  "type": "Create",
  "actor": "https://www.opendatahacklab.org/lap_src/actor.php?username=testmaster",
  "object": {
    "id": "https://www.opendatahacklab.org/lap_users/testmaster/status.json",
        "type": "Note",
        "published": "2023-06-09T16:40:00+02:00",
        "content": "Questo è uno stato di esempio",
        "tag": [
          {
            "type": "Mention",
            "href": "https://mastodon.social/users/cristianolongo",
            "name": "@cristianolongo@mastodon.social"
          }
        ]
  }
}
```

Esempio: replica


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://www.opendatahacklab.org/lap_users/testmaster/activity/create_reply",
  "type": "Create",
  "actor": "https://www.opendatahacklab.org/lap_src/actor.php?username=testmaster",
  "object": {
                "id": "https://www.opendatahacklab.org/lap_users/testmaster/reply",
                "type": "Note",
                "published": "2024-06-09T17:16:00+02:00",
                "inReplyTo": "https://mastodon.social/users/cristianolongo/statuses/112587287532450729",
                "content": "Che vuoi!?!?",
                 "tag": [
                   {
                     "type": "Mention",
                     "href": "https://mastodon.social/users/cristianolongo",
                     "name": "@cristianolongo@mastodon.social"
                   }
                 ]
  }
}
```

Esempio: modificare uno stato


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://www.opendatahacklab.org/lap_users/testmaster/activity/updstatus",
  "type": "Update",
  "actor": "https://www.opendatahacklab.org/lap_src/actor.php?username=testmaster",
  "object": {
    "id": "https://www.opendatahacklab.org/lap_users/testmaster/status.json",
        "type": "Note",
        "published": "2023-06-09T16:40:00+02:00",
        "updated": "2023-06-09T16:43:00+02:00",
        "content": "Questo è uno stato di esempio modificato",
        "tag": [
         {
           "type": "Mention",
           "href": "https://mastodon.social/users/cristianolongo",
           "name": "@cristianolongo@mastodon.social"
         }
       ]
  }
}
```

Esempio: eliminare uno stato


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://www.opendatahacklab.org/lap_users/testmaster/activity/17179387687",
  "type": "Delete",
  "actor": "https://www.opendatahacklab.org/lap_src/actor.php?username=testmaster",
  "object": "https://www.opendatahacklab.org/lap_users/testmaster/status.json"
}
```

La classe Collection

Le *collection* rappresentano elenchi di *Object* o *Link*

- Link
- **Object**
  - *Attori*
  - Activity
  - **Collection**
    - CollectionPage
    - OrderedCollection
  - *Contenuti*
  - ...

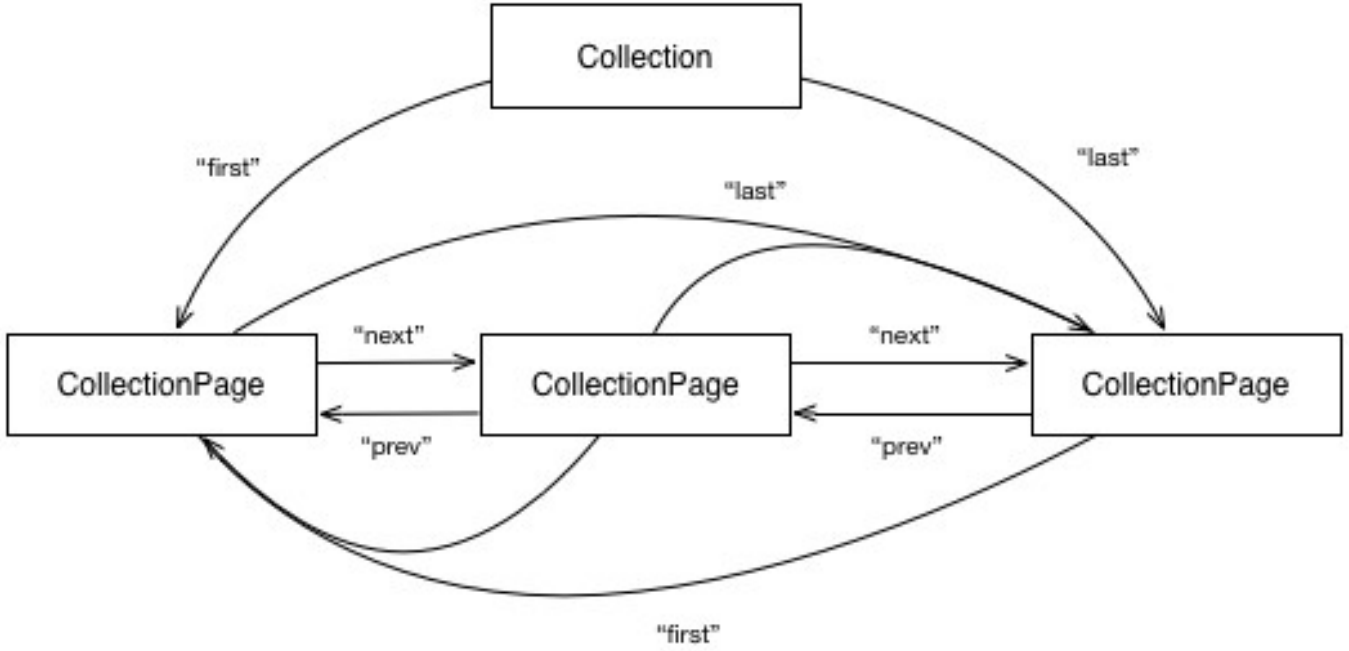
La proprietà items

Gli oggetti nella collection sono rappresentati con la proprietà *items*

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "summary": "Object history",
  "type": "Collection",
  "totalItems": 2,
  "items": [
    {
      "type": "Create",
      "actor": "http://www.test.example/sally",
      "object": "http://example.org/foo"
    },
    {
      "type": "Like",
      "actor": "http://www.test.example/joe",
      "object": "http://example.org/foo"
    }
  ]
}
```

Collection con paginazione

Gli elementi in una collection possono a loro volta essere raggruppati in *CollectionPage*.



Collection per gli attori

In ActivityPub ad ogni attore sono associate svariate collection

- Inbox
- Outbox
- Followers
- Followed
- Likes
- Liked
- ...

Collection per gli attori: inbox e outbox

In ActivityPub ad ogni attore sono associate svariate collection

- Inbox activity ricevute dall’attore
- Outbox activity inviate dall’attore
- ...

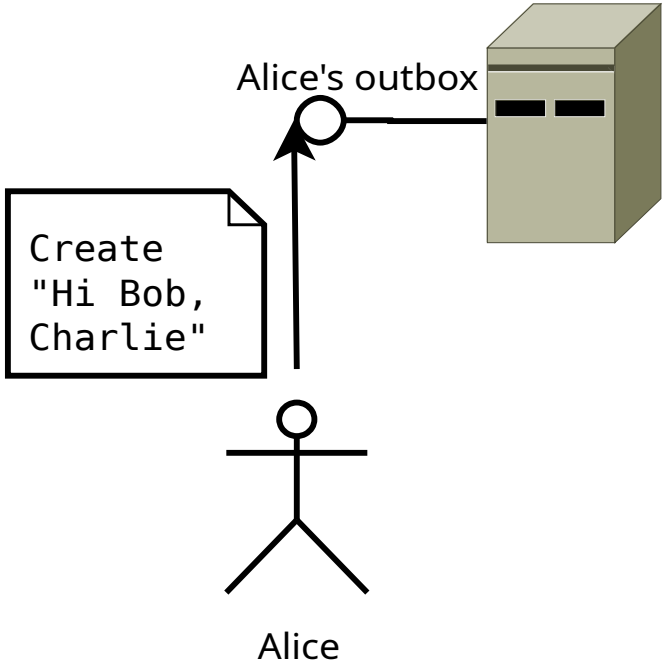
Collection per gli attori: followers e followed

Se Alice invia una Activity *Follow* a Bob e Bob accetta inviando una Activity *Accept* ad Alice, relativa alla follow request, allora

- Alice sarà aggiunta alla Collection *follower* di Bob,
- Bob sarà aggiunto alla Collection *followed* di Alice.

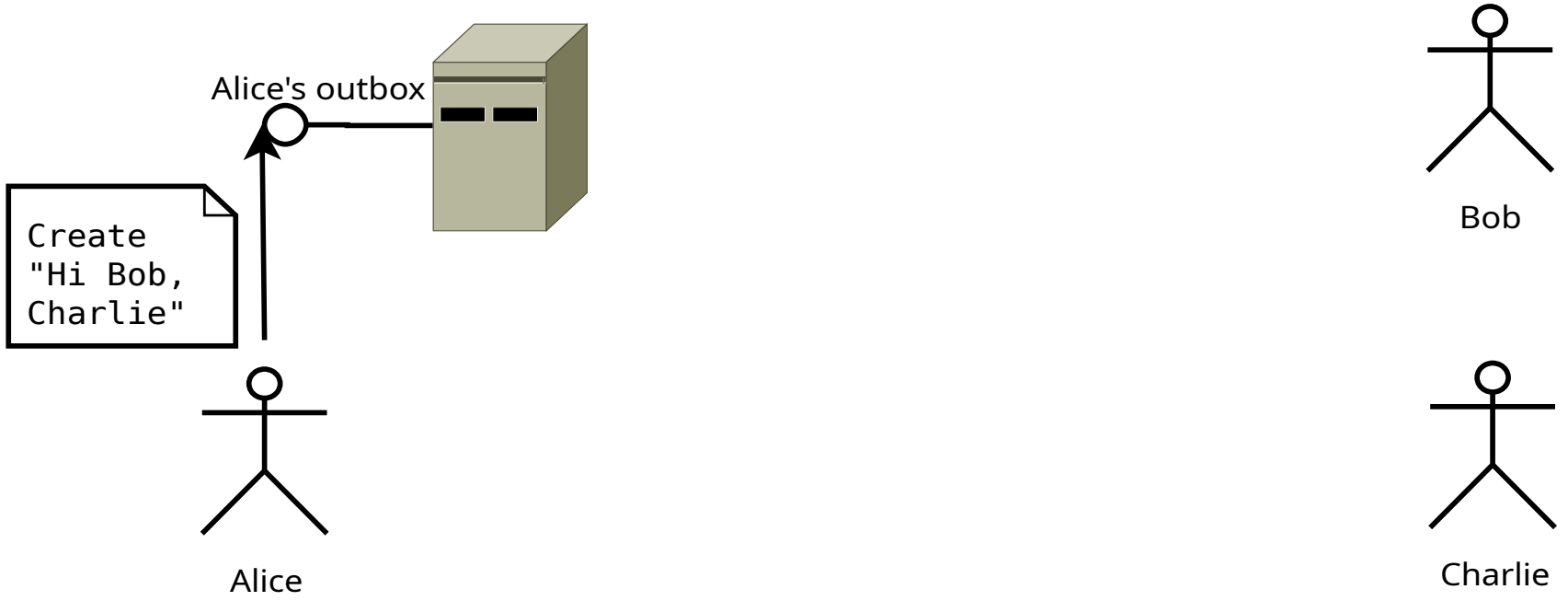
Invio delle activity: outbox

Client to Server: per inviare una activity, l’attore la pone nella propria outbox.



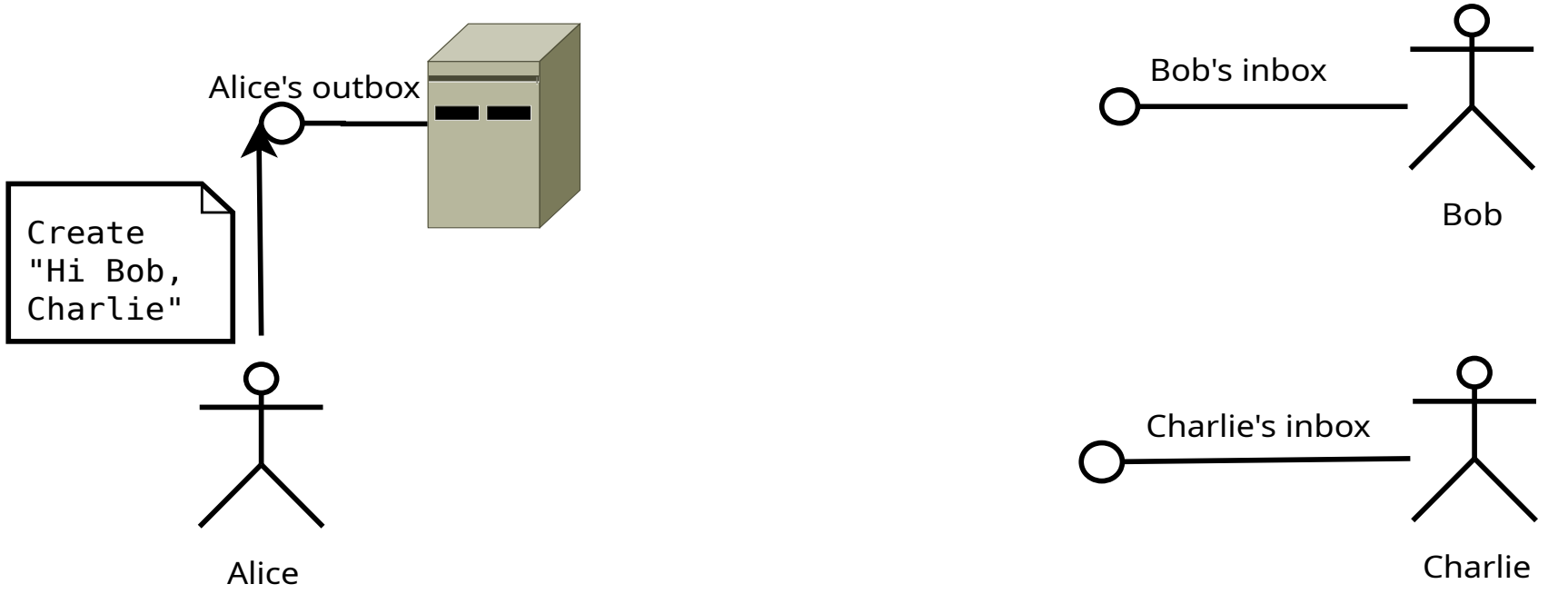
Invio delle activity: **audience targeting**

Il server determina gli attori a cui l'Activity deve essere inviata.



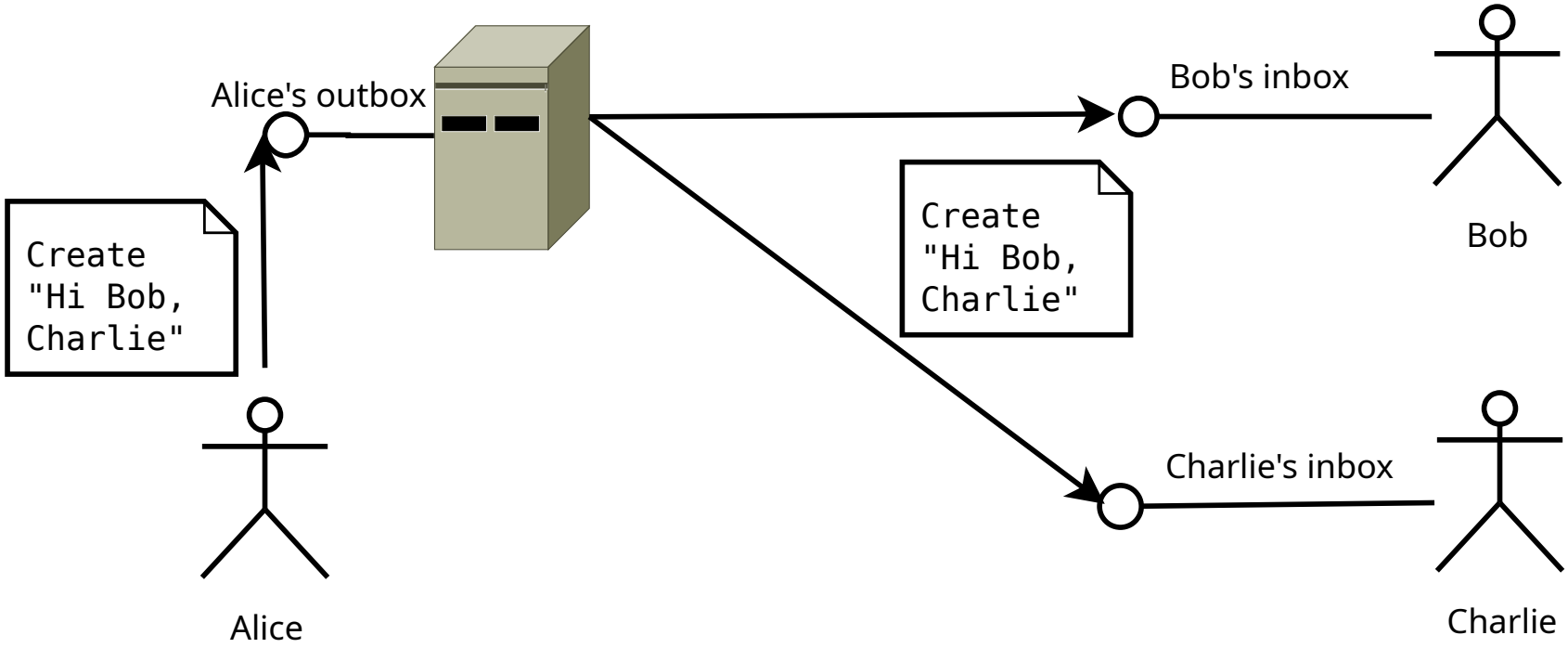
Invio delle activity: delivery (1/2)

Ottiene la inbox di ogni attore presente nell’audience dal suo profilo.



Invio delle activity: delivery (2/2)

Il server invia l’attività a queste inbox.



Audience Targeting

In ActivityPub, il server determina gli attori cui inviare una attività esaminando i campi **to**, **cc**, **bto**, **bcc** and **audience**.


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://www.opendatahacklab.org/lap_users/testmaster/activity/newstatustoaudiencetwo",
  "type": "Create",
  "actor": "https://www.opendatahacklab.org/lap_src/actor.php?username=testmaster",
  "to": "https://mastodon.social/users/cristianolongo",
  "object": {
    "id": "https://www.opendatahacklab.org/lap_users/testmaster/newstatustoaudiencetwo.json",
        "type": "Note",
        "published": "2023-06-10T11:09:00+02:00",
        "content": "Testare l'invio di un'altro stato con un audience",
        "tag": [
         {
           "type": "Mention",
           "href": "https://mastodon.social/users/cristianolongo",
           "name": "@cristianolongo@mastodon.social"
         }
       ]
  }
}
```

Collection e Audience Targeting

Nei campi **to**, **cc**, **bto**, **bcc** and **audience** possono essere presenti delle collection: tutti gli attori nella collection fanno parte dell'audience


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://www.opendatahacklab.org/lap_users/testmaster/activity/newstatustoaudiencetwo",
  "type": "Create",
  "actor": "https://www.opendatahacklab.org/lap_src/actor.php?username=testmaster",
  "to": "https://mastodon.social/users/cristianolongo",
  "cc": "https://www.opendatahacklab.org/testmasterfollowers",
  "object": {
    "id": "https://www.opendatahacklab.org/lap_users/testmaster/newstatustoaudiencetwo.json",
        "type": "Note",
        "published": "2023-06-10T11:09:00+02:00",
        "content": "Testare l'invio di un'altro stato con un audience",
        "tag": [
         {
           "type": "Mention",
           "href": "https://mastodon.social/users/cristianolongo",
           "name": "@cristianolongo@mastodon.social"
         }
       ]
  }
}
```

Contenuti *pubblici*

Se in **to** o **cc** è presente https://www.w3.org/ns/activitystreams#Public, il contenuto è considerato *pubblico.*


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://www.opendatahacklab.org/lap_users/testmaster/activity/newstatustoaudiencetwo",
  "type": "Create",
  "actor": "https://www.opendatahacklab.org/lap_src/actor.php?username=testmaster",
  "to": "https://mastodon.social/users/cristianolongo",
  “cc”: "https://www.w3.org/ns/activitystreams#Public”,
  "object": {
    "id": "https://www.opendatahacklab.org/lap_users/testmaster/newstatustoaudiencetwo.json",
        "type": "Note",
        "published": "2023-06-10T11:09:00+02:00",
        "content": "Testare l'invio di un'altro stato con un audience",
        "tag": [
         {
           "type": "Mention",
           "href": "https://mastodon.social/users/cristianolongo",
           "name": "@cristianolongo@mastodon.social"
         }
       ]
  }
}
```

Delivery: sharedInbox

Per un attore può essere definita una *sharedInbox*

```
{
      "@context": [
            "https://www.w3.org/ns/activitystreams","https://w3id.org/security/v1", {    }],
      "id": "https://mastodon.bida.im/users/aaronwinstonsmith",
      "type": "Person",
…
      "endpoints": {
            "sharedInbox": "https://mastodon.bida.im/inbox"
      },
...
}
```

Delivery: sharedInbox

Per un attore può essere definita una *sharedInbox*

Se due o più attori condividono la stessa sharedInbox, è sufficiente inviare le attività su questa.


```
{
    "@context": [
        "https://www.w3.org/ns/activitystreams","https://w3id.org/security/v1", {    }],
    "id": "https://mastodon.bida.im/users/aaronwinstonsmith",
    "type": "Person",
…
    "endpoints": {
        "sharedInbox": "https://mastodon.bida.im/inbox"
    },
...
}
```

Delivery: sharedInbox

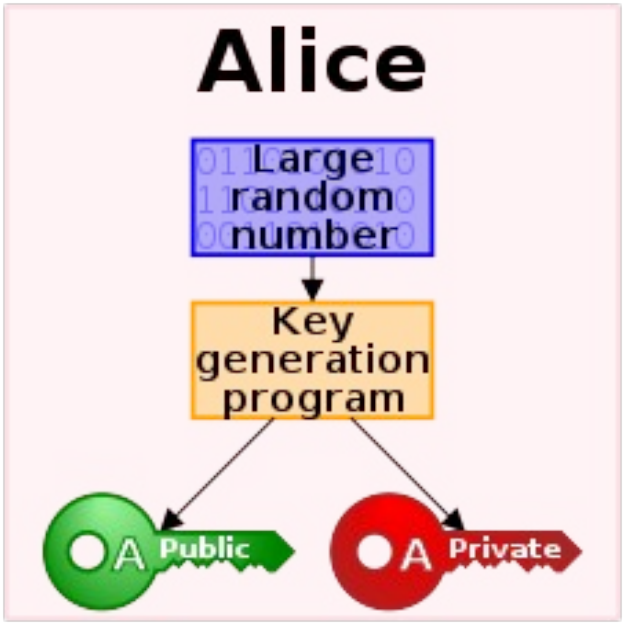
Per un attore può essere definita una *sharedInbox*

Se due o più attori condividono la stessa sharedInbox, è sufficiente inviare le attività su questa.

```
{
    "@context": [
        "https://www.w3.org/ns/activitystreams","https://w3id.org/security/v1", {    }],
    "id": "https://mastodon.bida.im/users/aaronwinstonsmith",
    "type": "Person",
…
    "endpoints": {
        "sharedInbox": "https://mastodon.bida.im/inbox"
    },
...
}
```

È la shared inbox di tutti gli utenti di mastodon.bida.im

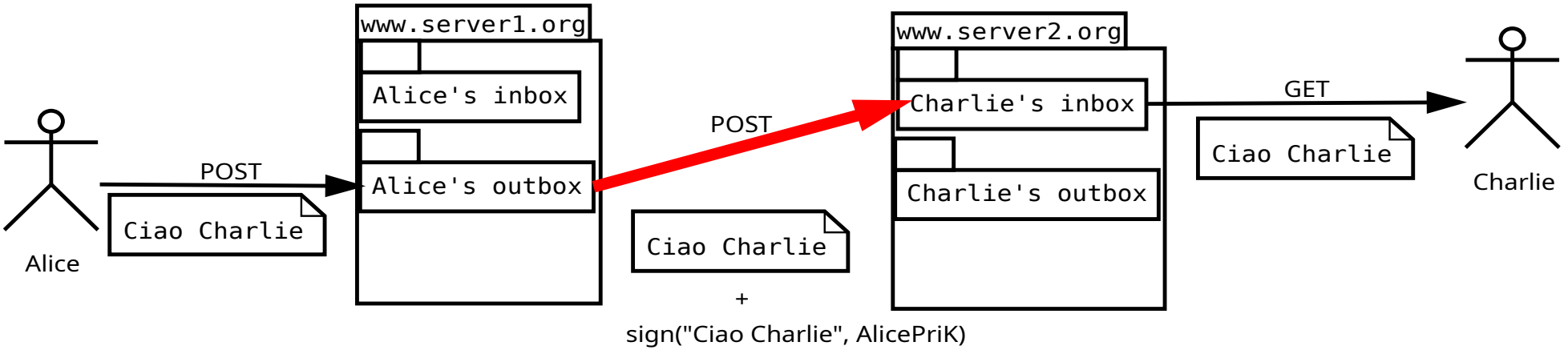
HTTP Signatures



Autenticazione Server to Server

Per inviare una attività alla inbox di un altro utente è necessario che la richiesta sia firmata con HTTP Signatures.

https://datatracker.ietf.org/doc/html/draft-cavage-http-signatures



Chiavi crittografiche

La chiave pubblica deve essere indicata nella descrizione dell'attore usando il vocabolario https://w3id.org/security/v1.

```
{
    "@context": [
        "https://www.w3.org/ns/activitystreams","https://w3id.org/security/v1", {    }],
    "id": "https://mastodon.bida.im/users/aaronwinstonsmith",
    "type": "Person",
    …,
    "publicKey": {
        "id": "https://mastodon.bida.im/users/aaronwinstonsmith#main-key",
        "owner": "https://mastodon.bida.im/users/aaronwinstonsmith",
        "publicKeyPem": "-----BEGIN PUBLIC KEY-----\n ...\n-----END PUBLIC KEY-----\n"
    }, ...
}
```

Chiavi crittografiche

ATTENZIONE: in publicKeyPem sostituire gli “a capo” con “\n”

```
{
    "@context": [
        "https://www.w3.org/ns/activitystreams","https://w3id.org/security/v1", {    }],
    "id": "https://mastodon.bida.im/users/aaronwinstonsmith",
    "type": "Person",
    …,
    "publicKey": {
        "id": "https://mastodon.bida.im/users/aaronwinstonsmith#main-key",
        "owner": "https://mastodon.bida.im/users/aaronwinstonsmith",
        "publicKeyPem": "-----BEGIN PUBLIC KEY-----\n ...\n-----END PUBLIC KEY-----\n"
    }, ...
}
```

HTTP Signatures

HTTP Signatures: aggiungere un header *Signature* contenente la firma di alcuni altri header

keyId è la URL specificata come publicKey nell'attore

Signature : **keyId**="https://mastodon.bida.im/users/aaronwinstonsmith#main-key",algorithm="rsa-sha256",headers="(request-target) host date digest content-type",signature="…."

HTTP Signatures

HTTP Signatures: aggiungere un header *Signature* contenente la firma di alcuni altri header

keyId è la URL specificata come publicKey nell'attore

algorithm è l'algoritmo di firma

Signature : keyId="https://mastodon.bida.im/users/aaronwinstonsmith#main-key",**algorithm**="rsa-sha256",headers="(request-target) host date digest content-type",signature="…."

HTTP Signatures

HTTP Signatures: aggiungere un header *Signature* contenente la firma di alcuni altri header

keyId è la URL specificata come publicKey nell'attore

algorithm è l'algoritmo di firma

headers è la lista di header HTTP che saranno firmati (request-target è fittizio, digest è obbligatorio)

Signature : keyId="https://mastodon.bida.im/users/aaronwinstonsmith#main-key",algorithm="rsa-sha256",**headers**="(request-target) host date digest content-type",signature="…."

HTTP Signatures – generazione della firma

Innanzitutto è necessario che tra gli header ci sia il *digest* dell’attività.

```
$activityDigest='SHA-256='.base64_encode(openssl_digest($activity, 'SHA256', true));
```

HTTP Signatures – generazione della firma

La stringa da firmare è costituita da una riga per ogni header, del formato

<header name lowercased>: <header value>

(request-target): post users/aaronwinstonsmith/inbox
host: mastodon.bida.im
date: Thu, 23 Feb 2023 17:46:31 GMT
digest: $activityDigest

HTTP Signatures – generazione della firma

La stringa da firmare è costituita da una riga per ogni header, del formato

<header name lowercased>: <header value>

(request-target): post users/aaronwinstonsmith/inbox
host: mastodon.bida.im
date: Thu, 23 Feb 2023 17:46:31 GMT
digest: $activityDigest

Questo messaggio va firmato con la chiave privata dell'attore.

```
openssl_sign($toBeSigned, $signature, $privatekey, OPENSSL_ALGO_SHA256);
```

The End
(grazie a tutti)