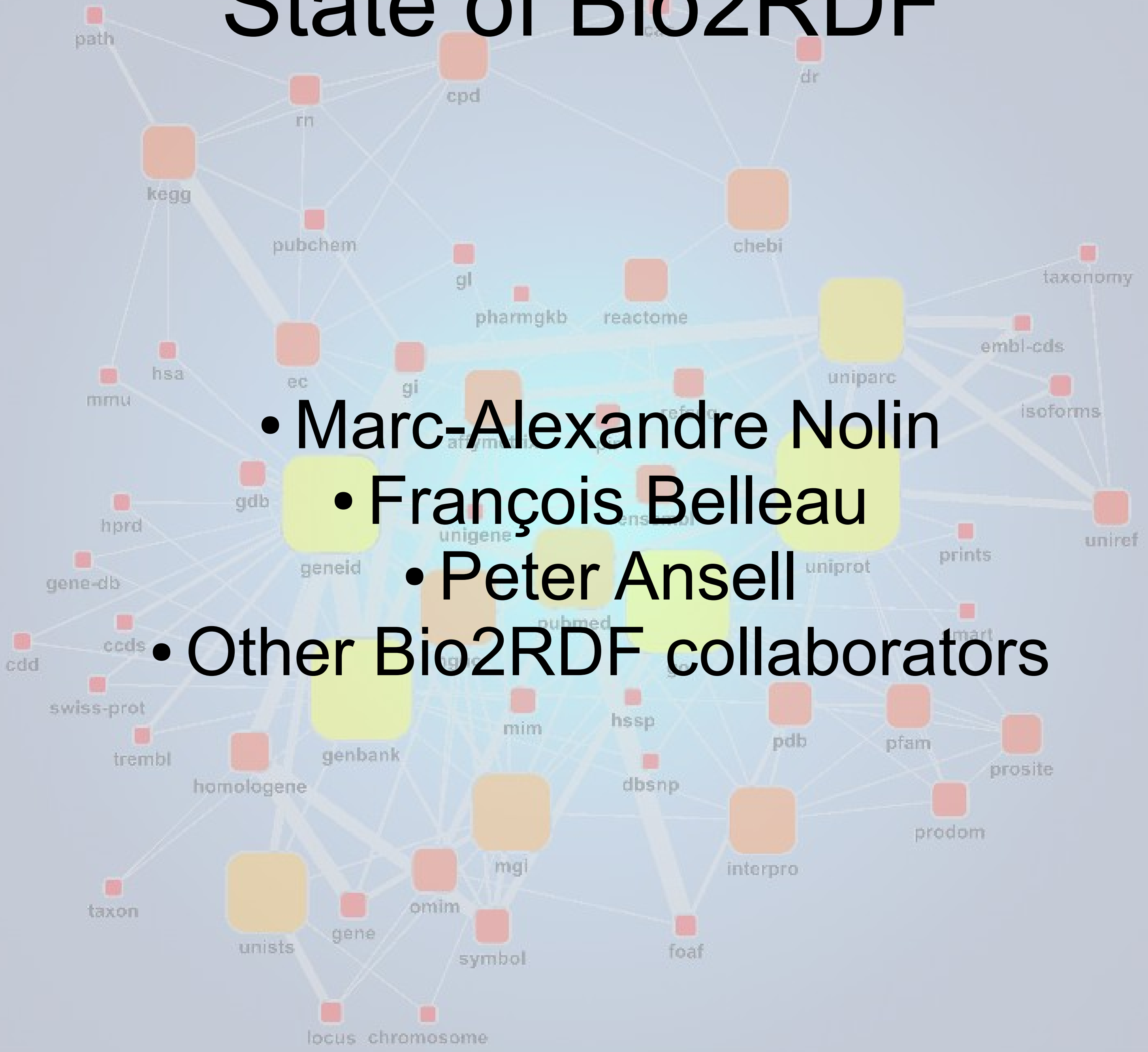
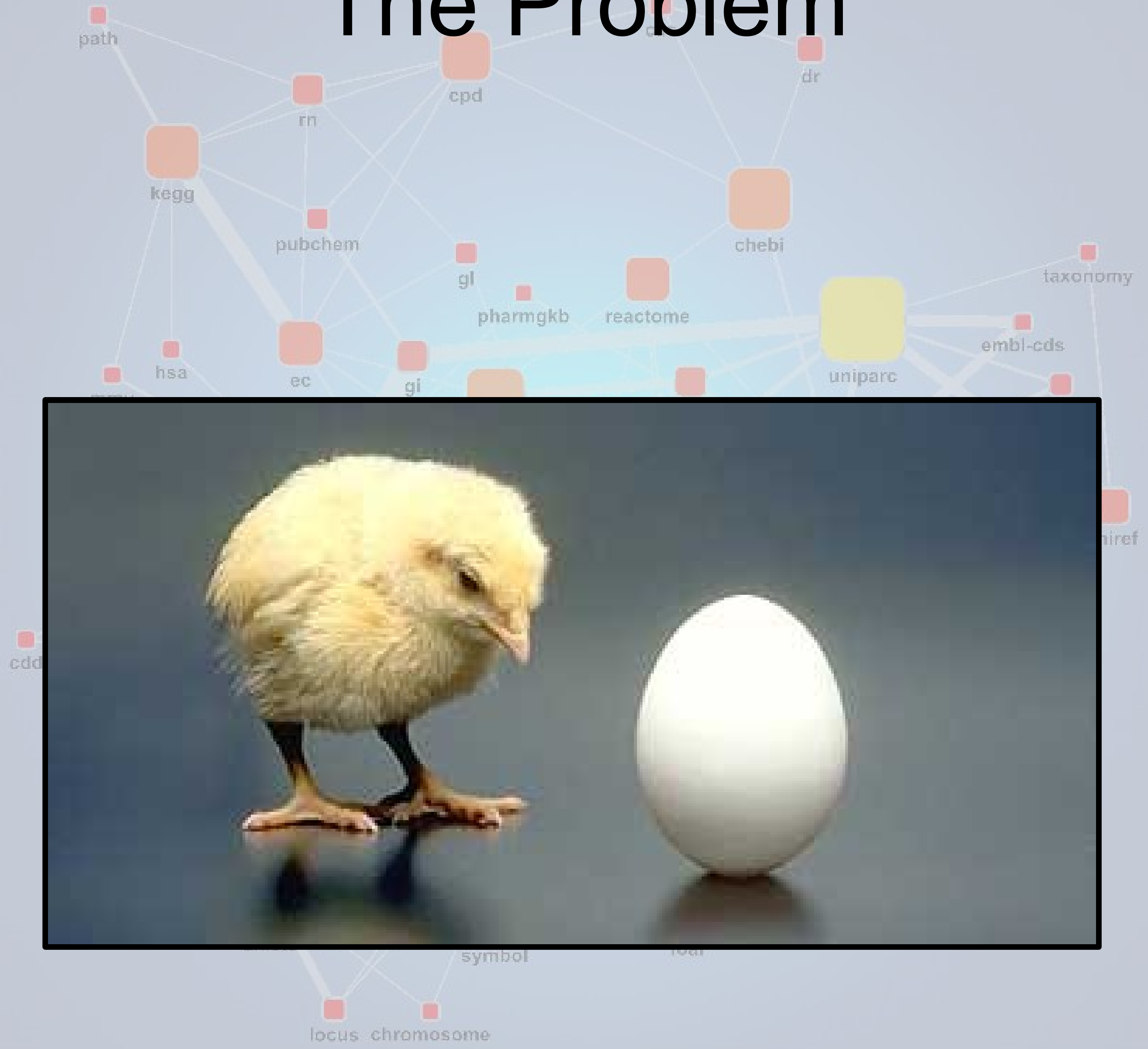
State of Bio2RDF

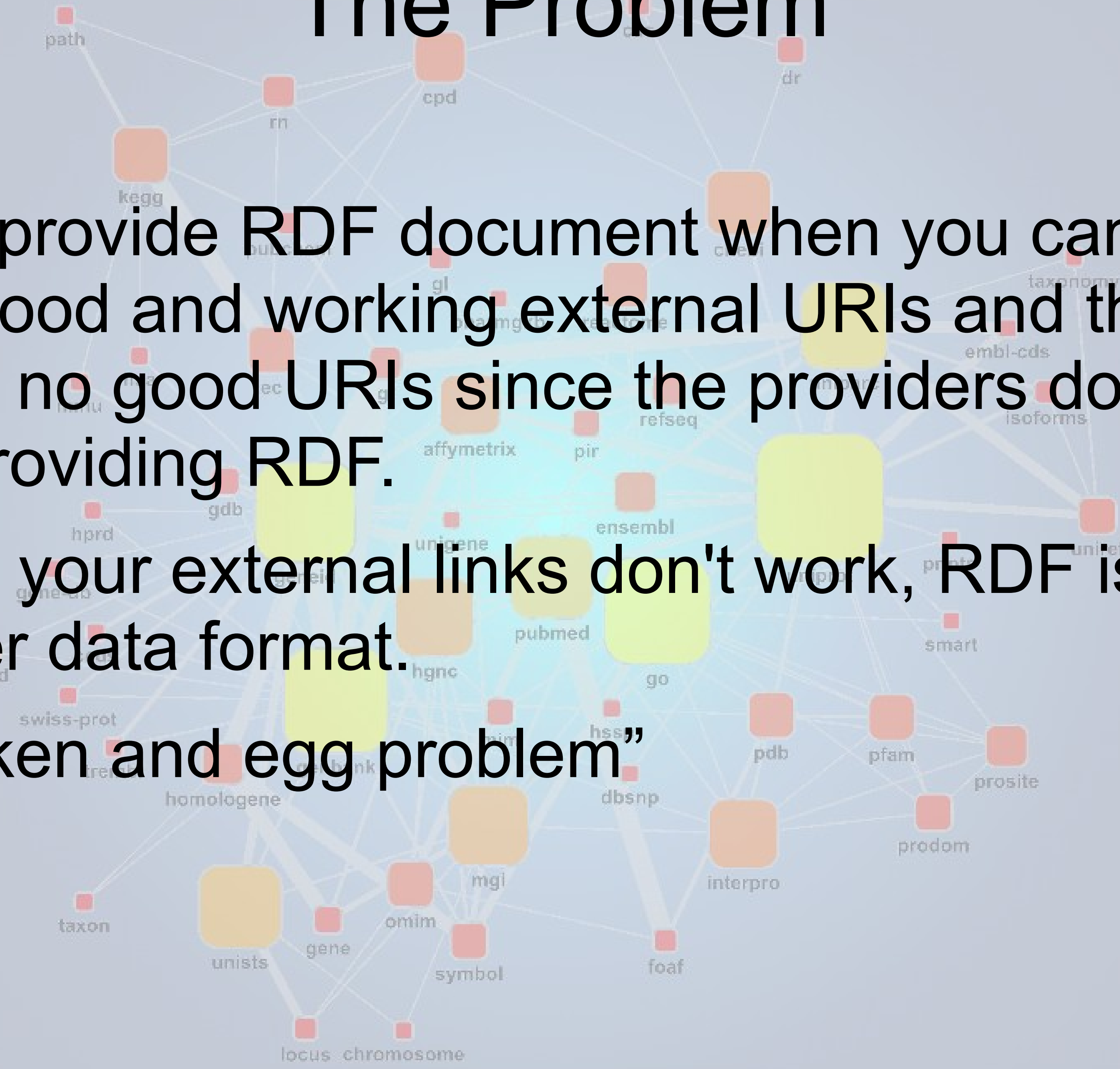


- Marc-Alexandre Nolin
- François Belleau
- Peter Ansell
- Other Bio2RDF collaborators

The Problem

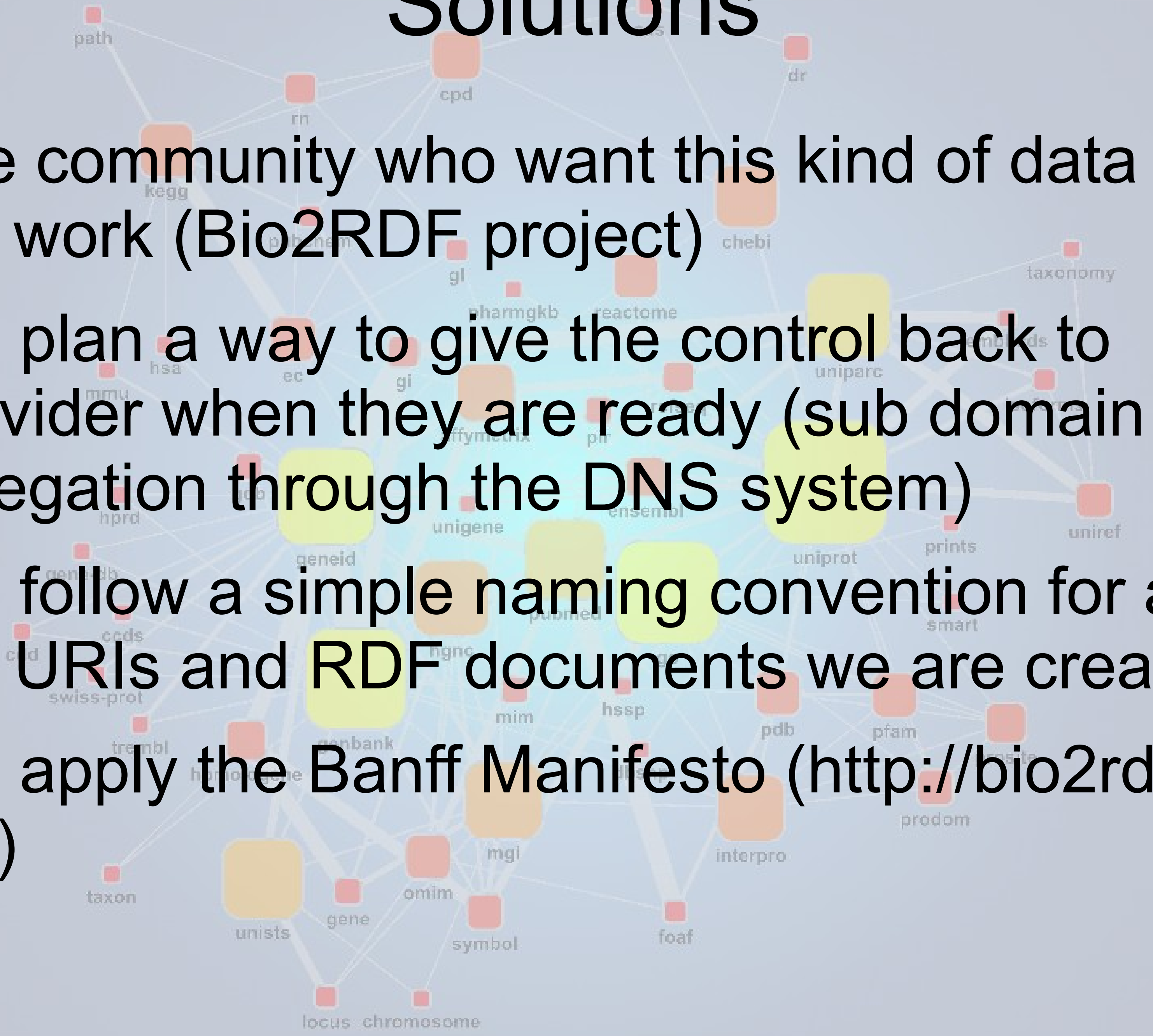

path
cpd
dr
rn
kegg
chebi
pubchem
gl
taxonomy
pharmgkb
reactome
embl-cds
hsa
ec
gi
uniparc
cdd
symbol
locus
chromosome

The Problem



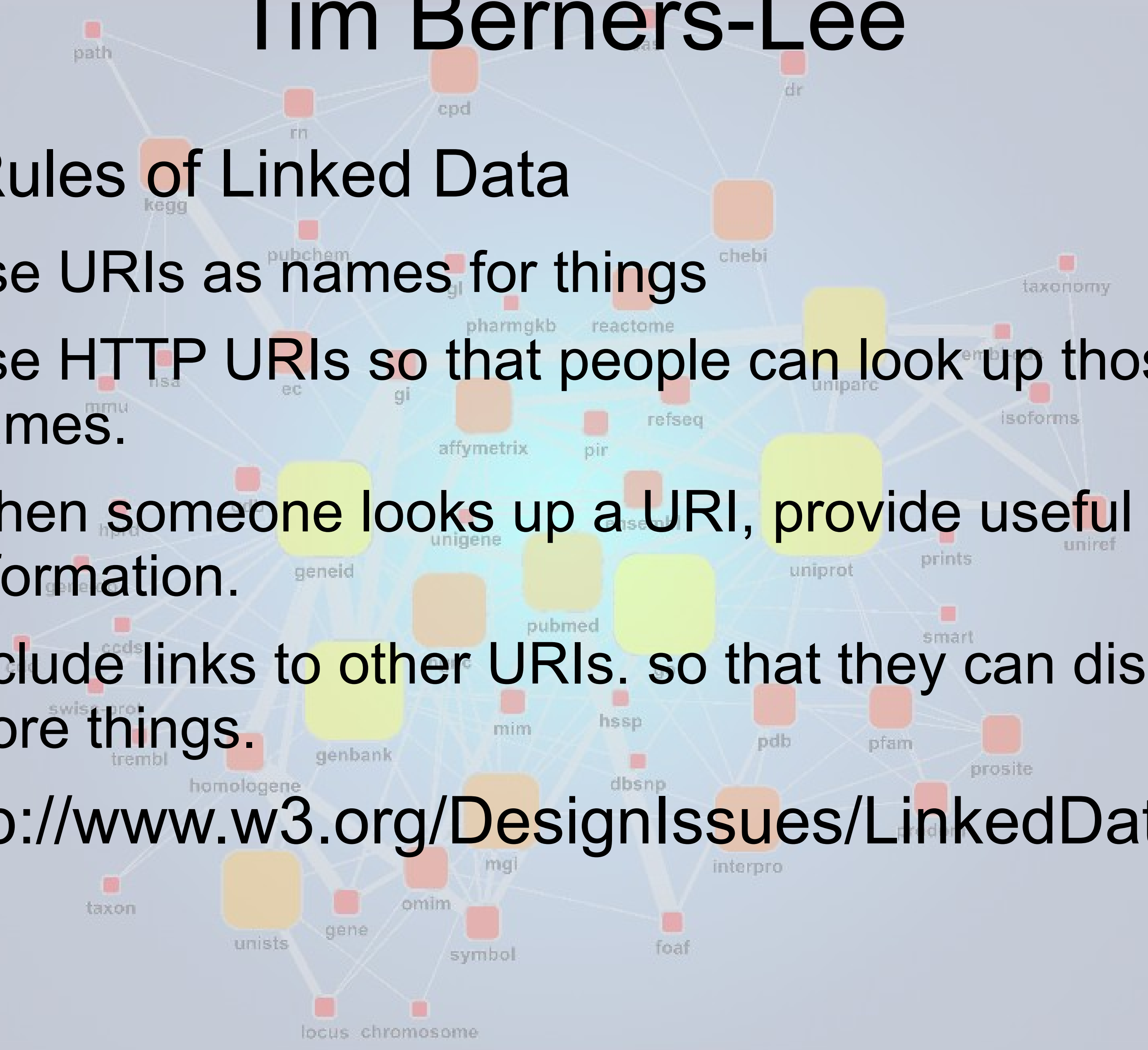
- How provide RDF document when you can not write good and working external URIs and there is almost no good URIs since the providers do not start providing RDF.
- When your external links don't work, RDF is just another data format.
- “Chicken and egg problem”

Solutions



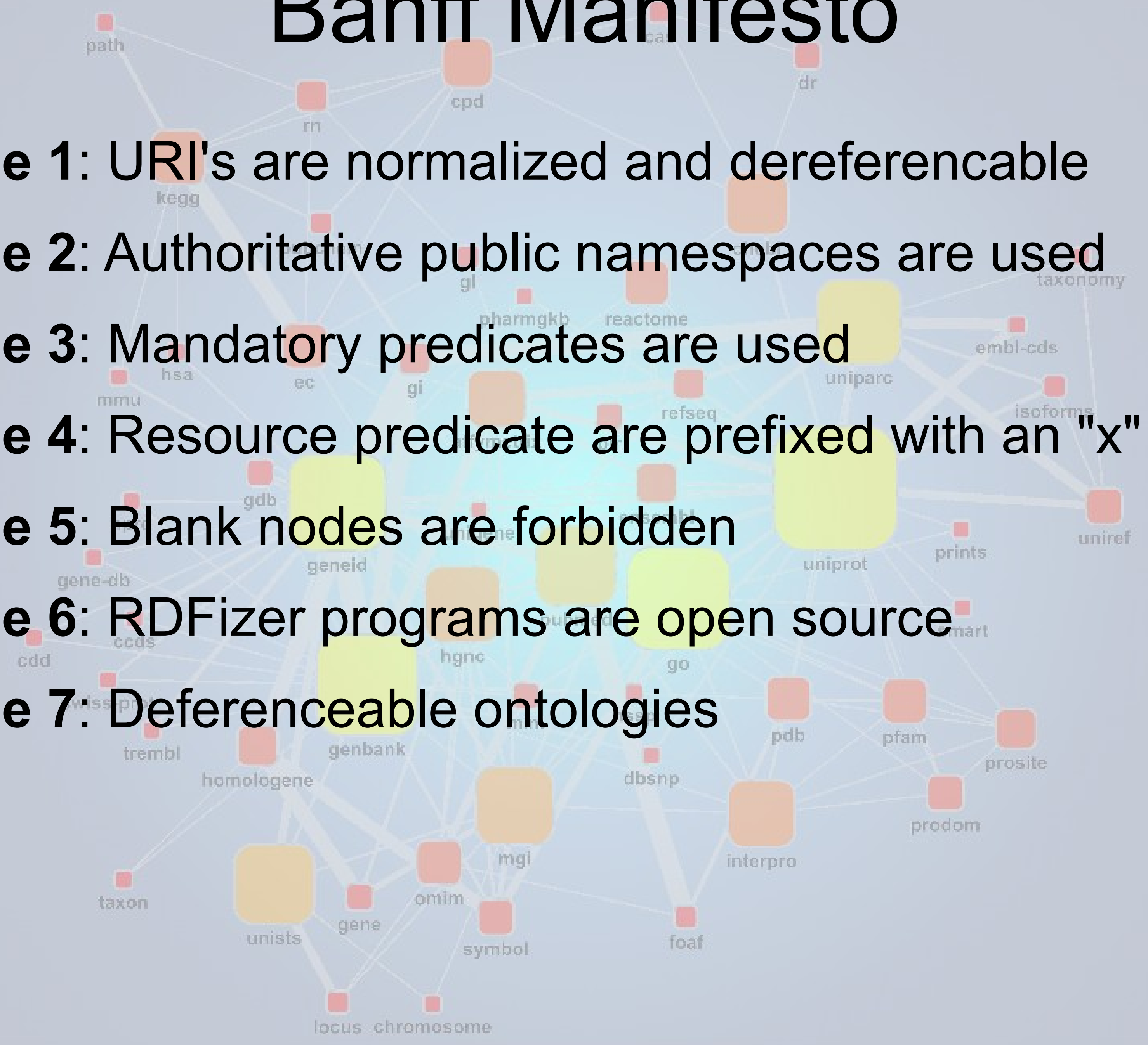
- The community who want this kind of data do the work (Bio2RDF project)
- We plan a way to give the control back to provider when they are ready (sub domain delegation through the DNS system)
- We follow a simple naming convention for all the URIs and RDF documents we are creating.
- We apply the Banff Manifesto (http://bio2rdf.org/bm)

Tim Berners-Lee



- 4 Rules of Linked Data
  - Use URIs as names for things
  - Use HTTP URIs so that people can look up those names.
  - When someone looks up a URI, provide useful information.
  - Include links to other URIs. so that they can discover more things.
- http://www.w3.org/DesignIssues/LinkedData

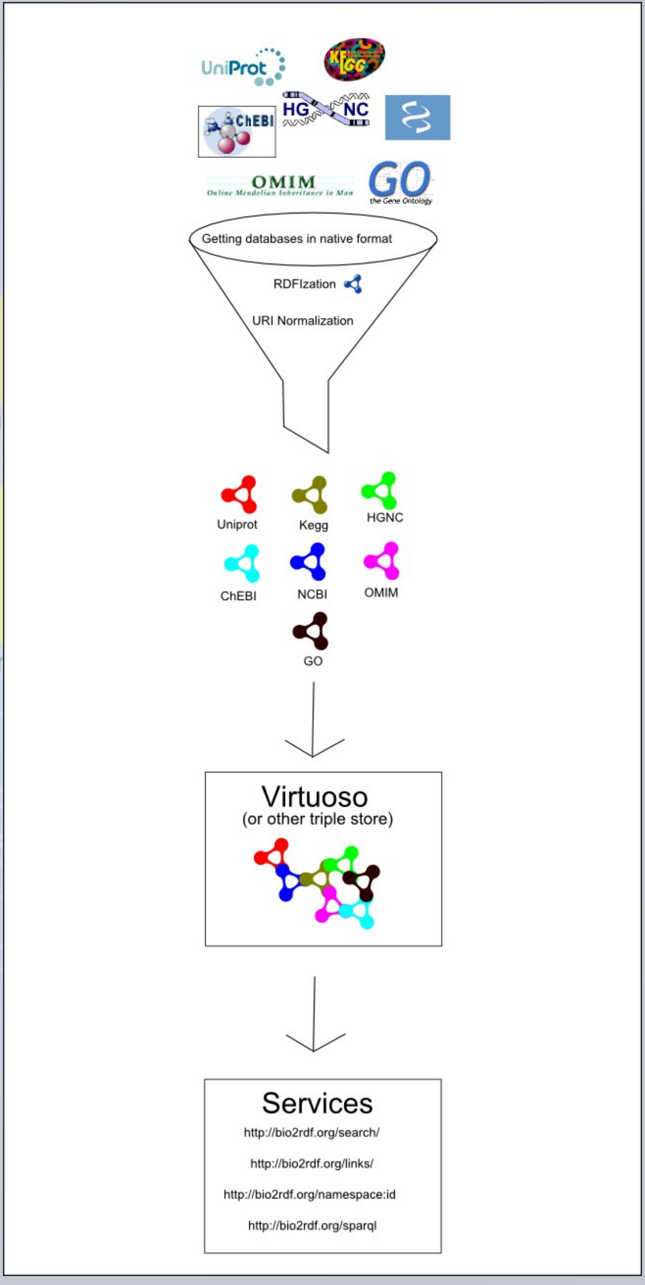
Banff Manifesto



- **Rule 1**: URI's are normalized and dereferencable
- **Rule 2**: Authoritative public namespaces are used
- **Rule 3**: Mandatory predicates are used
- **Rule 4**: Resource predicate are prefixed with an "x"
- **Rule 5**: Blank nodes are forbidden
- **Rule 6**: RDFizer programs are open source
- **Rule 7**: Deferenceable ontologies

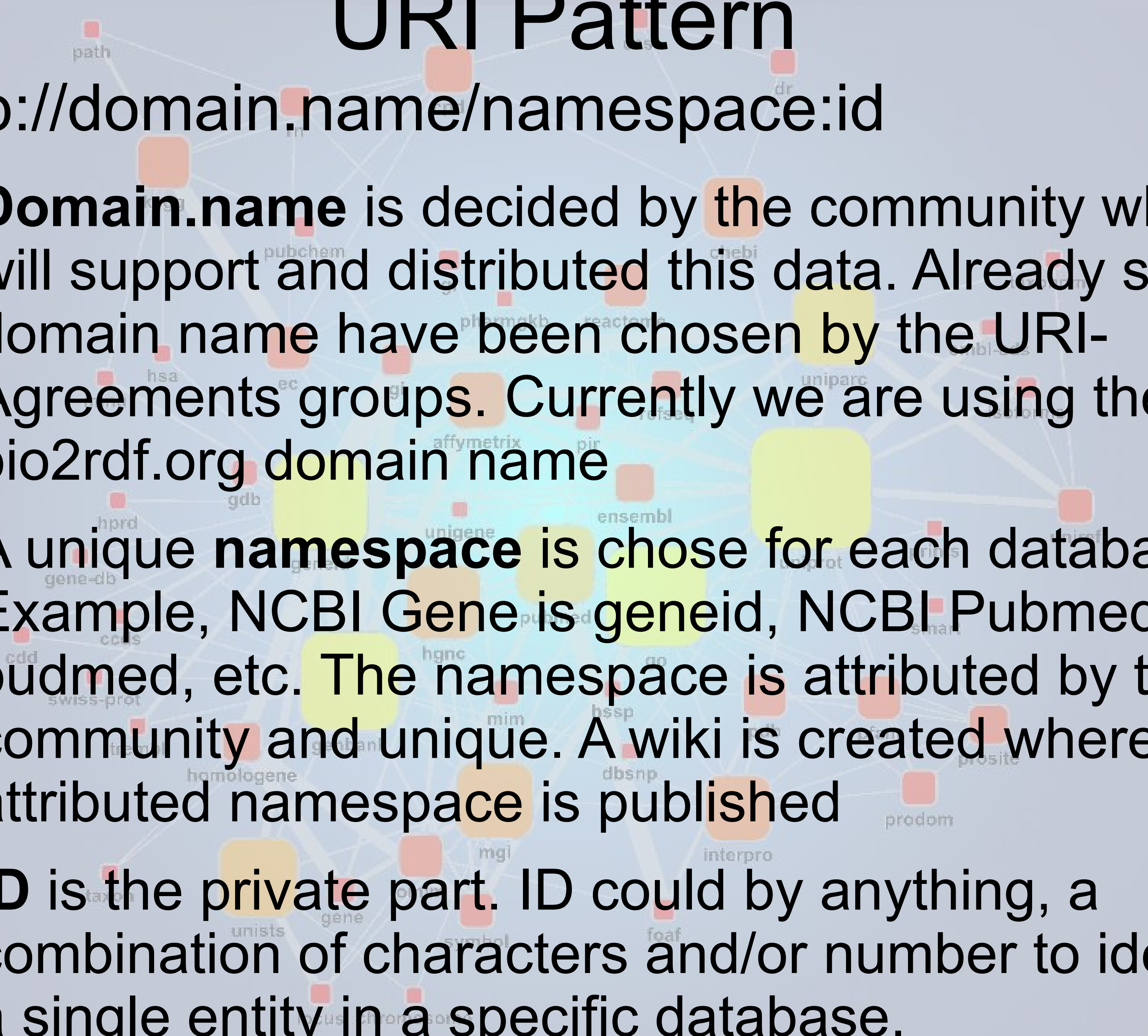
Bio2RDF Process

- Downloading the original data
- Creating a script (perl, jsp, xsl, etc.) to transform the original data into RDF
- Keeping a link to the original data in the new RDF document
- Store the converted database in a triple store. We currently use Virtuoso
- Give services for these using a RESTful interface (document retrieval, search, reverse links)

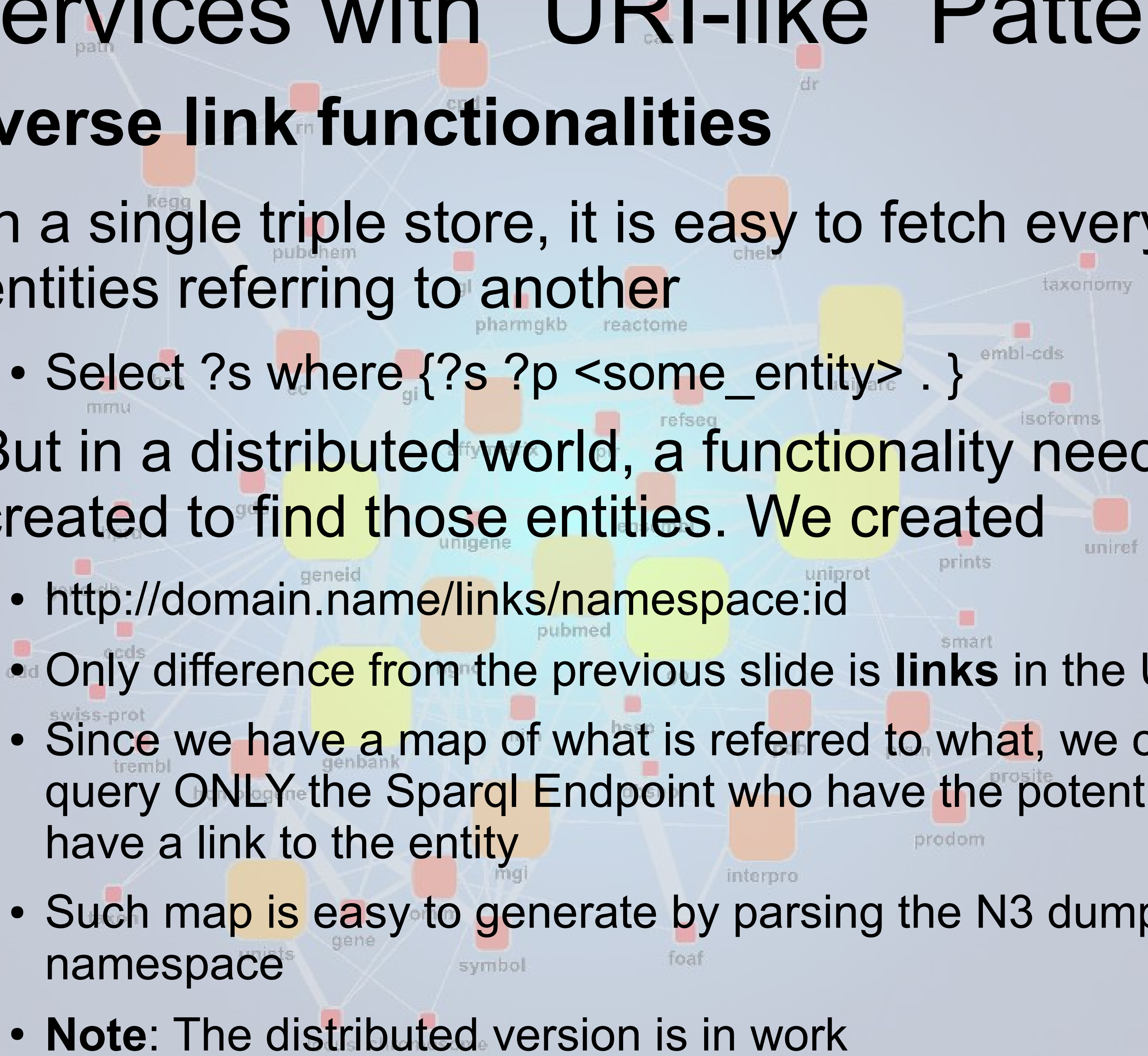


UniProt
KEGG
ChEBI
HGNC
OMIM
Online Mendelian Inheritance in Man
GO
the Gene Ontology
Getting databases in native format
RDFIzation
URI Normalization
Uniprot
Kegg
HGNC
ChEBI
NCBI
OMIM
GO
Virtuoso
(or other triple store)
Services
http://bio2rdf.org/search/
http://bio2rdf.org/links/
http://bio2rdf.org/namespace:id
http://bio2rdf.org/sparql

URI Pattern

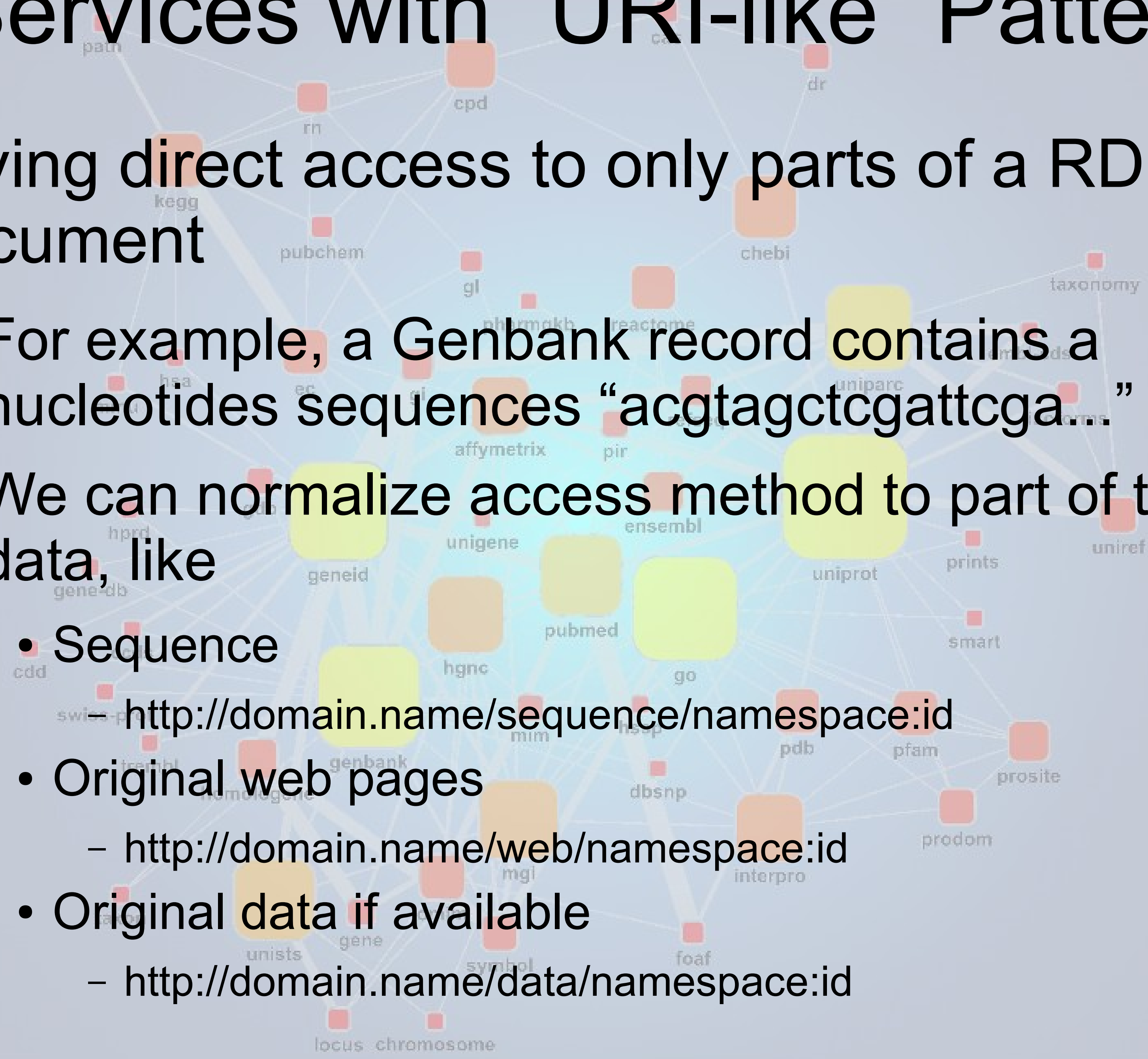


- http://domain.name/namespace:id
  - **Domain.name** is decided by the community who will support and distributed this data. Already some domain name have been chosen by the URI-Agreements groups. Currently we are using the bio2rdf.org domain name
  - A unique **namespace** is chose for each databases. Example, NCBI Gene is geneid, NCBI Pubmed is pudmed, etc. The namespace is attributed by the community and unique. A wiki is created where the attributed namespace is published
  - **ID** is the private part. ID could by anything, a combination of characters and/or number to identify a single entity in a specific database.



Services with “URI-like” Pattern

- **Reverse link functionalities**
  - In a single triple store, it is easy to fetch every entities referring to another
    - Select ?s where {?s ?p <some_entity> . }
  - But in a distributed world, a functionality need to be created to find those entities. We created
    - http://domain.name/links/namespace:id
    - Only difference from the previous slide is **links** in the URI
    - Since we have a map of what is referred to what, we can query ONLY the Sparql Endpoint who have the potential to have a link to the entity
    - Such map is easy to generate by parsing the N3 dump of a namespace
    - **Note**: The distributed version is in work



Services with “URI-like” Pattern

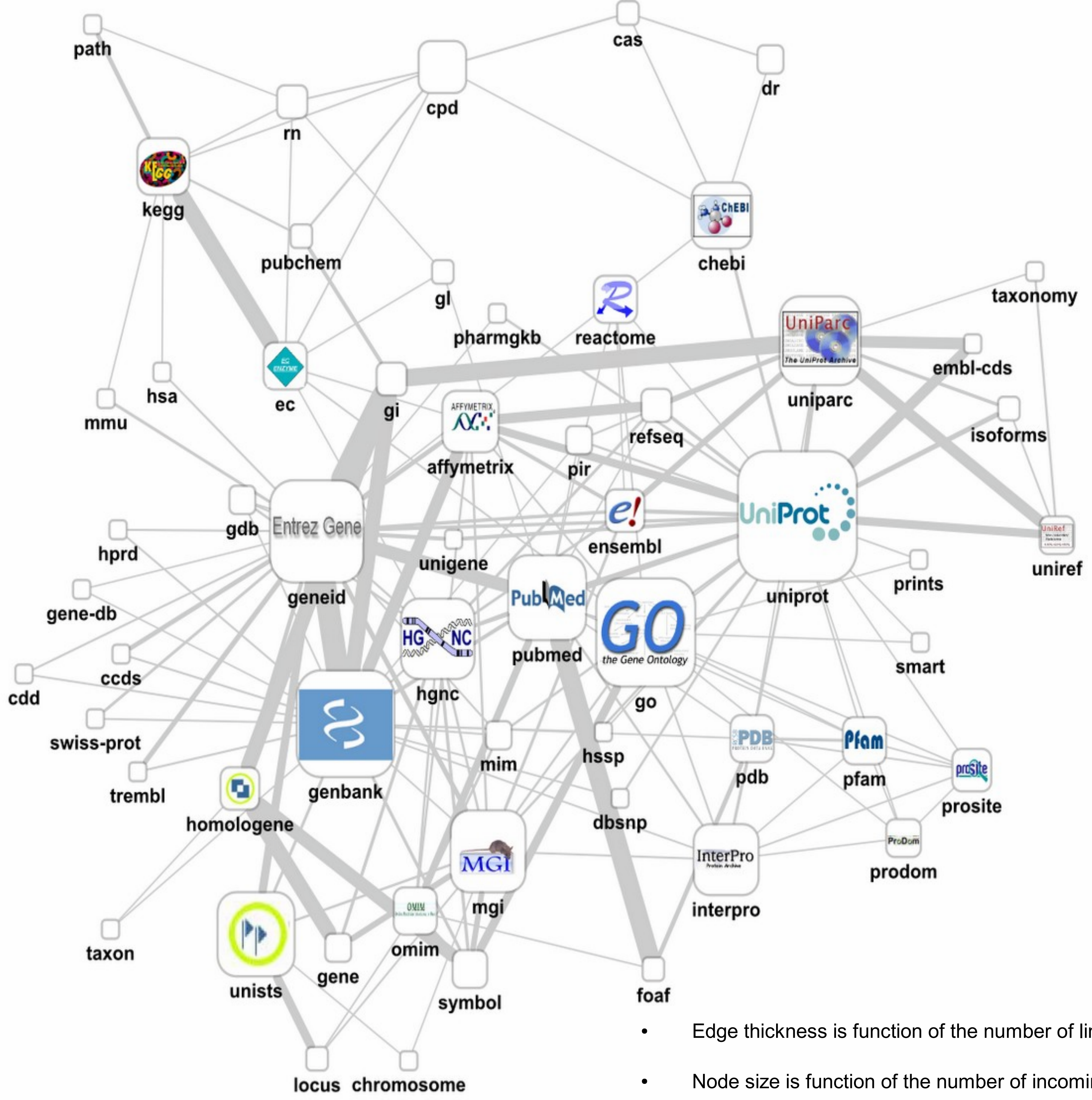
- Giving direct access to only parts of a RDF document
  - For example, a Genbank record contains a nucleotides sequences “acgtagctcgattcga...”
  - We can normalize access method to part of the data, like
    - Sequence
      - http://domain.name/sequence/namespace:id
    - Original web pages
      - http://domain.name/web/namespace:id
    - Original data if available
      - http://domain.name/data/namespace:id

Services with “URI-like” Pattern

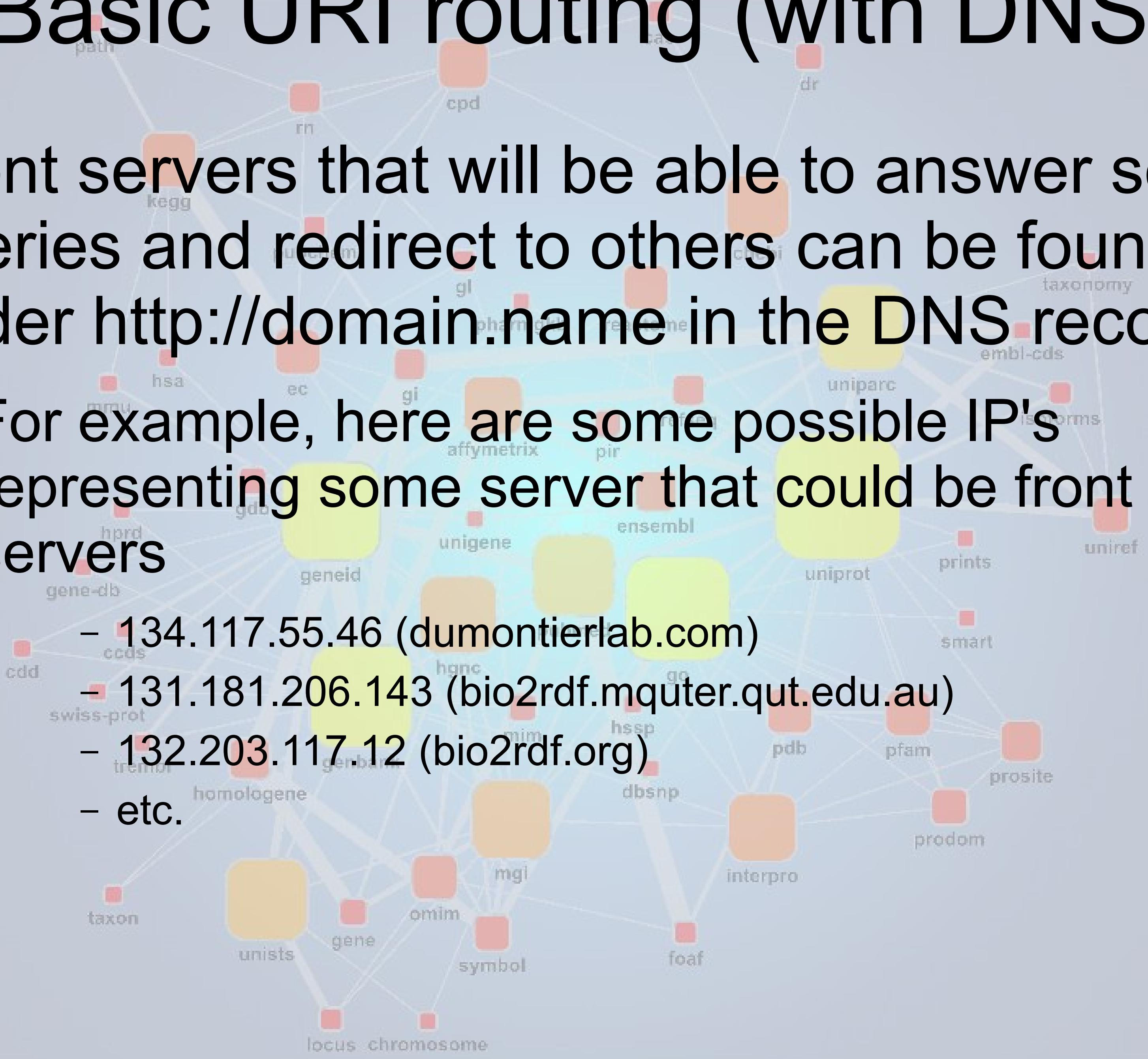
- Image if available (pathway have image for example)
  - http://domain.name/image/namespace:id
- These “URI-like” pattern can hide Sparql queries like these
  - Select ?web_page where{<http://domain.name/namespace:id> <http://domain.name/ns/genbank#WebPage> ?web_page}
  - This query only return the address of the original web page of the data, but the complete record is still in RDF, but hided to the user
  - If there is a generic pattern for a given namespace a rule can also be included in the package to dynamically generate the web page URL and redirect to it if needed

Services with “URI-like” Pattern

- **Full text Search**
  - We are currently using a triple store, Virtuoso, who is offering a full text search capability.
    - Select ?s where {?s ?p ?o . ?o bif:contains “_text_” .}
    - We can launch this query in parallel to the Sparql Endpoint and wait for the answer.
    - Granted, this could be longer to answer than the reverse links operator, but it could bring interesting result
    - http://domain.name/search/_text_
    - Such services is already in use in Bio2RDF.org on the repository about mouse ad human located at http://bio2rdf.org/sparql
    - A database specific search can also be produced using /searchns/database:searchTerm



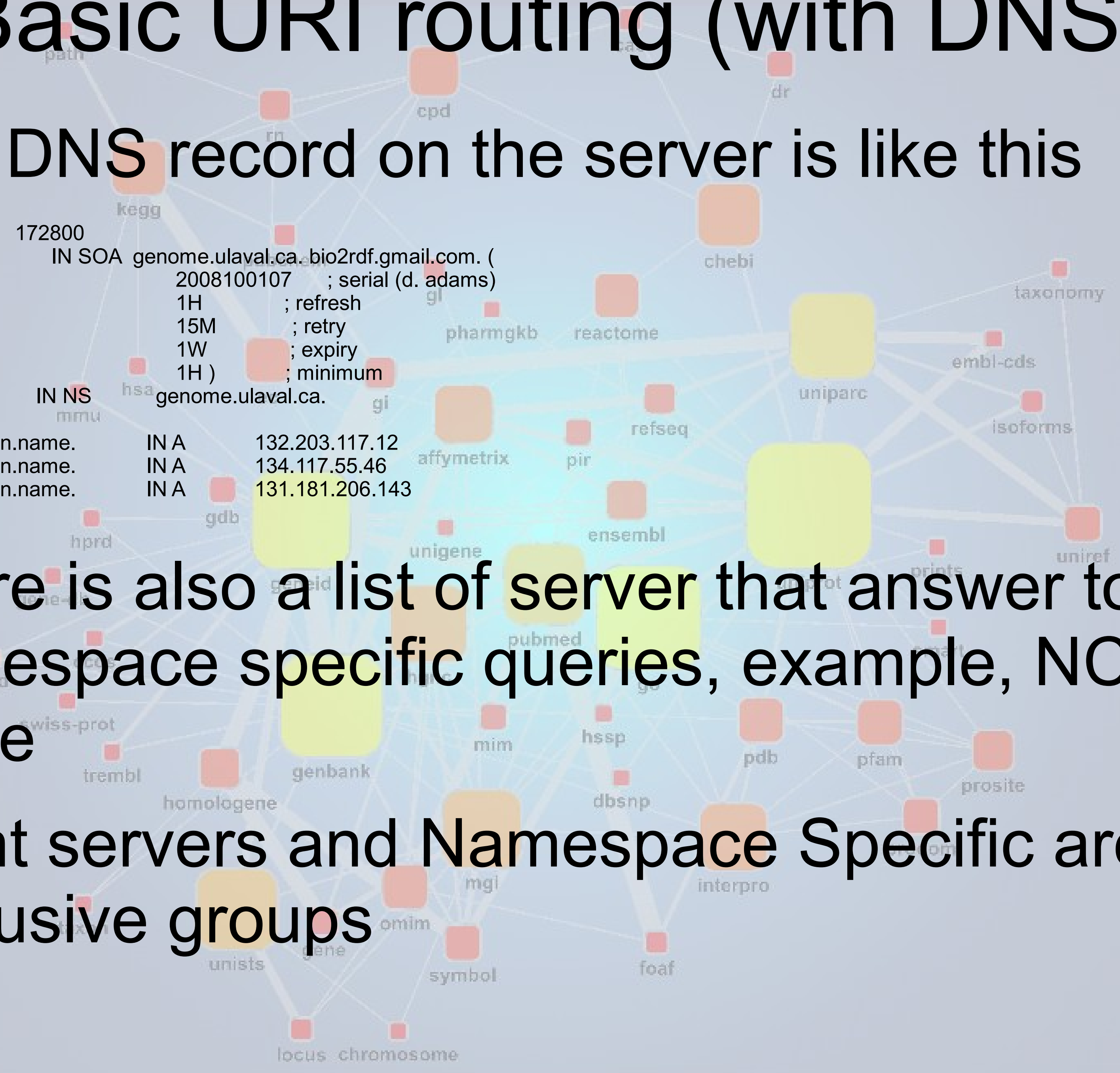
- Edge thickness is function of the number of links between the 2 nodes
- Node size is function of the number of incoming edge to the node



Basic URI routing (with DNS)

- Front servers that will be able to answer some queries and redirect to others can be found under http://domain.name in the DNS record.
  - For example, here are some possible IP's representing some server that could be front servers
    - 134.117.55.46 (dumontierlab.com)
    - 131.181.206.143 (bio2rdf.mquter.qut.edu.au)
    - 132.203.117.12 (bio2rdf.org)
    - etc.

Basic URI routing (with DNS)



- The DNS record on the server is like this

  -
    ```
    $TTL    172800
    @              IN SOA  genome.ulaval.ca. bio2rdf.gmail.com. (
                                2008100107      ; serial (d. adams)
                                1H              ; refresh
                                15M             ; retry
                                1W              ; expiry
                                1H )            ; minimum
                   IN NS       genome.ulaval.ca.

    domain.name.        IN A        132.203.117.12
    domain.name.        IN A        134.117.55.46
    domain.name.        IN A        131.181.206.143
    ...
    ```
- There is also a list of server that answer to namespace specific queries, example, NCBI Gene
- Front servers and Namespace Specific are not exclusive groups

Basic URI routing (with DNS)

- For example, here are a list of servers that answer to queries for the the NCBI Gene namespace
  - 132.203.117.12 (bio2rdf.org)
  - 29.148.159.201 (hcls.sciencecommons.org)
  - 131.181.206.143 (bio2rdf.mquter.qut.edu.au)
  - Etc.
- The DNS record on the server is like this

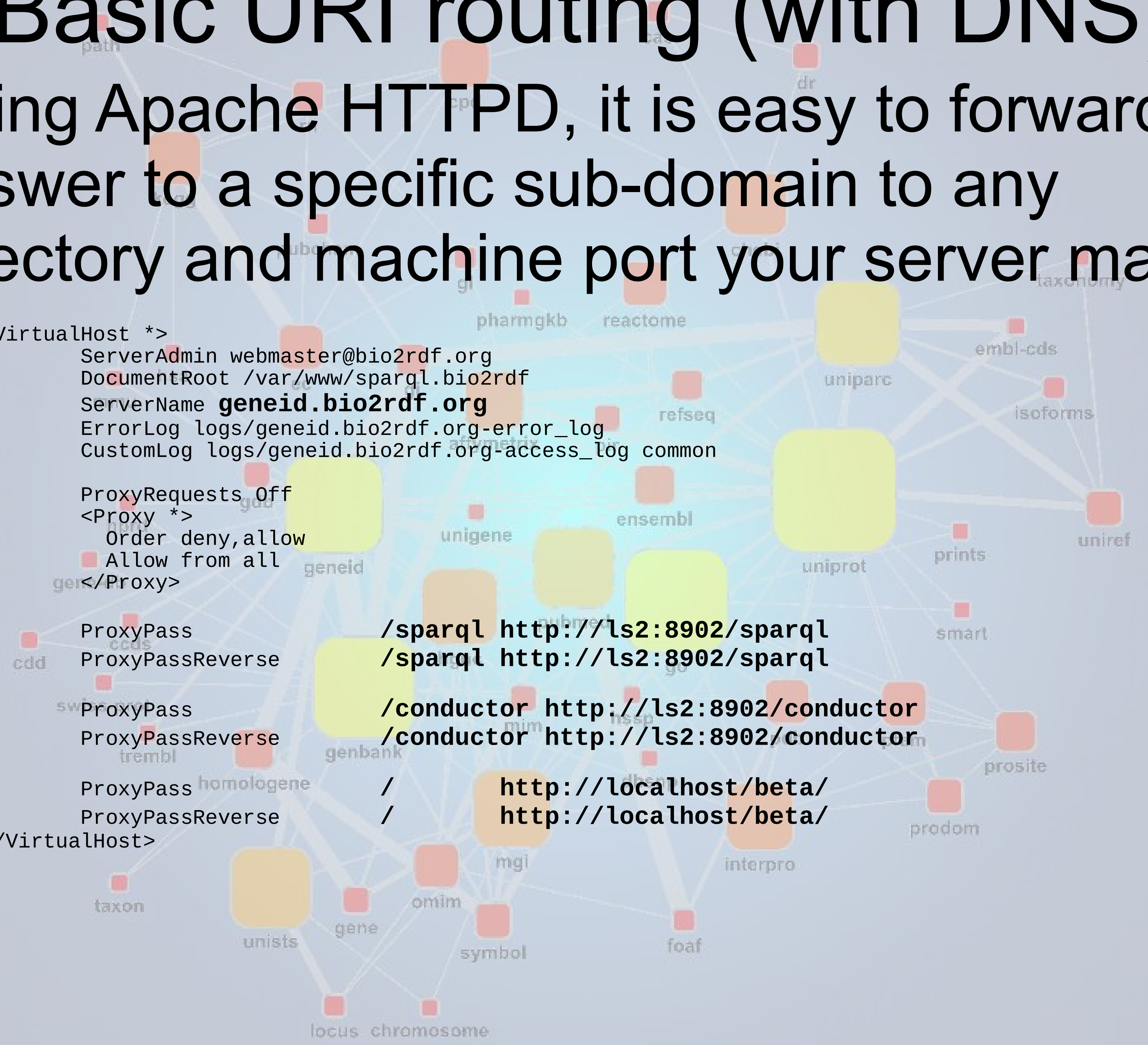
```
–   $TTL    172800
    @             IN SOA  genome.ulaval.ca. bio2rdf.gmail.com. (
                              2008100107      ; serial (d. adams)
                              1H              ; refresh
                              15M             ; retry
                              1W              ; expiry
                              1H )            ; minimum
                IN NS          genome.ulaval.ca.

    geneid.domain.name.        IN A           132.203.117.12
    geneid.domain.name.        IN A           29.148.159.201
```

Basic URI routing (with DNS)

- Like you see, there is no problem for a main server to also be able to answer to answer to some namespaces. In the example, 132.203.117.12 (bio2rdf.org) can answer to both.
- Giving subdomain is free to all who want to dedicate resources to provide linked data version of data for any namespaces
  - A caveat is that every provider of a specific namespaces needs to provide the same URI formatted data, with standard predicates such as dc:title and rdfs:label as minimum standards.

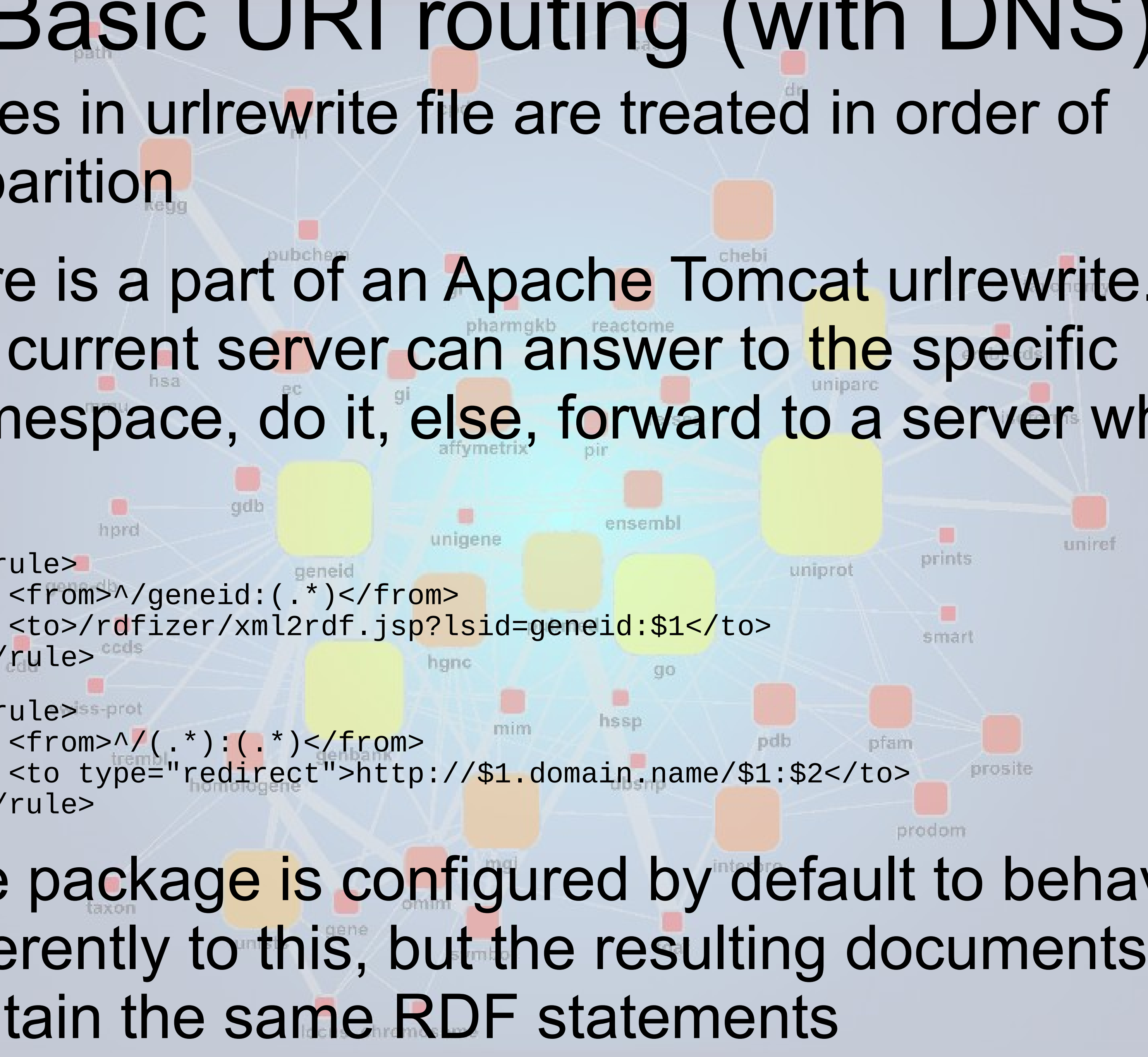
Basic URI routing (with DNS)

- Using Apache HTTPD, it is easy to forward the answer to a specific sub-domain to any directory and machine port your server may be.



–
```
<VirtualHost *>
        ServerAdmin webmaster@bio2rdf.org
        DocumentRoot /var/www/sparql.bio2rdf
        ServerName geneid.bio2rdf.org
        ErrorLog logs/geneid.bio2rdf.org-error_log
        CustomLog logs/geneid.bio2rdf.org-access_log common

        ProxyRequests Off
        <Proxy *>
          Order deny,allow
          Allow from all
        </Proxy>

        ProxyPass               /sparql http://ls2:8902/sparql
        ProxyPassReverse        /sparql http://ls2:8902/sparql

        ProxyPass               /conductor http://ls2:8902/conductor
        ProxyPassReverse        /conductor http://ls2:8902/conductor

        ProxyPass               /         http://localhost/beta/
        ProxyPassReverse        /         http://localhost/beta/
</VirtualHost>
```

Basic URI routing (with DNS)



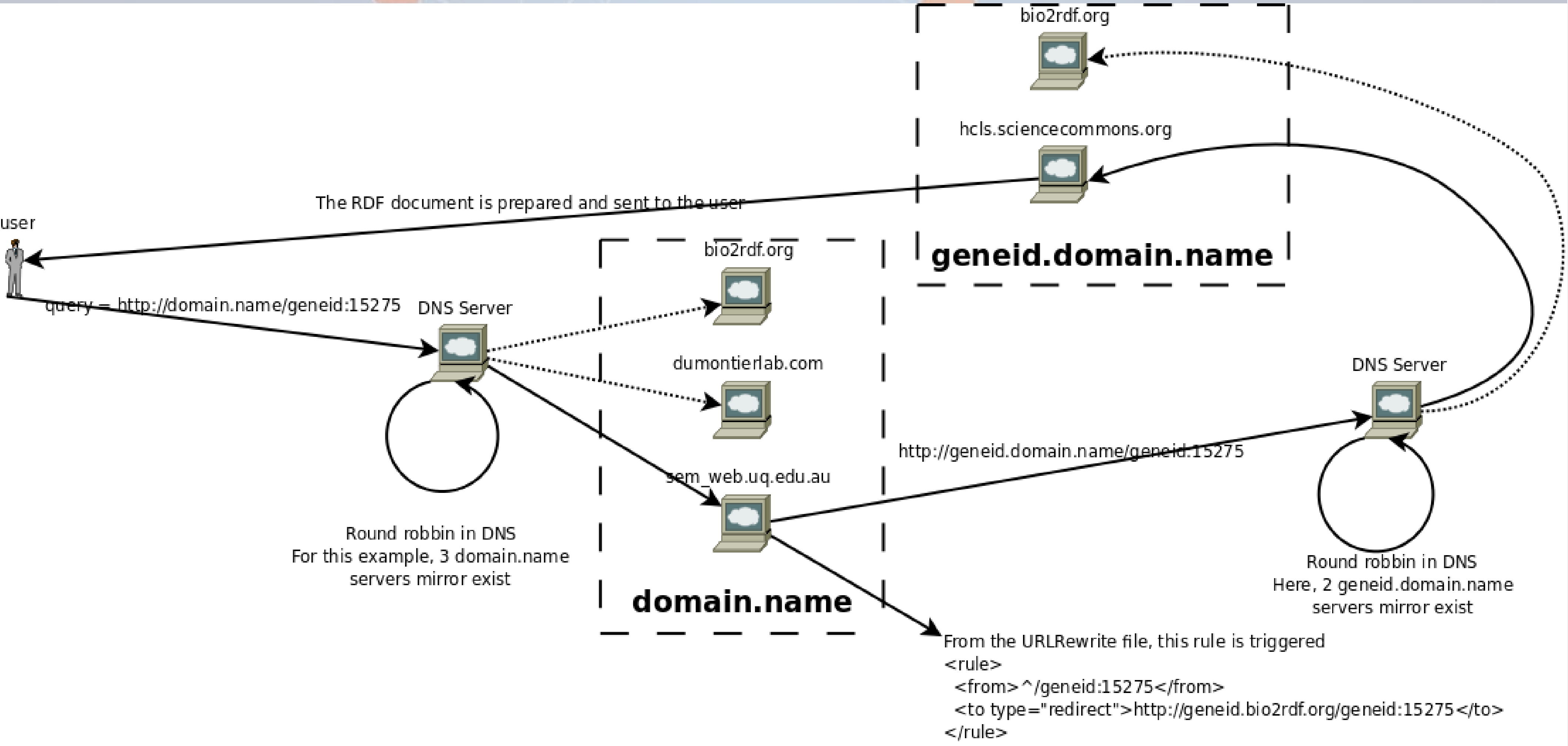
- Rules in urlrewrite file are treated in order of apparition
- Here is a part of an Apache Tomcat urlrewrite. If the current server can answer to the specific namespace, do it, else, forward to a server who can

```
– <rule>
    <from>^/geneid:(.*)</from>
    <to>/rdfizer/xml2rdf.jsp?lsid=geneid:$1</to>
  </rule>

– <rule>
    <from>^/(.*):(.*)</from>
    <to type="redirect">http://$1.domain.name/$1:$2</to>
  </rule>
```

- The package is configured by default to behave differently to this, but the resulting documents contain the same RDF statements

Basic URI routing (with DNS)

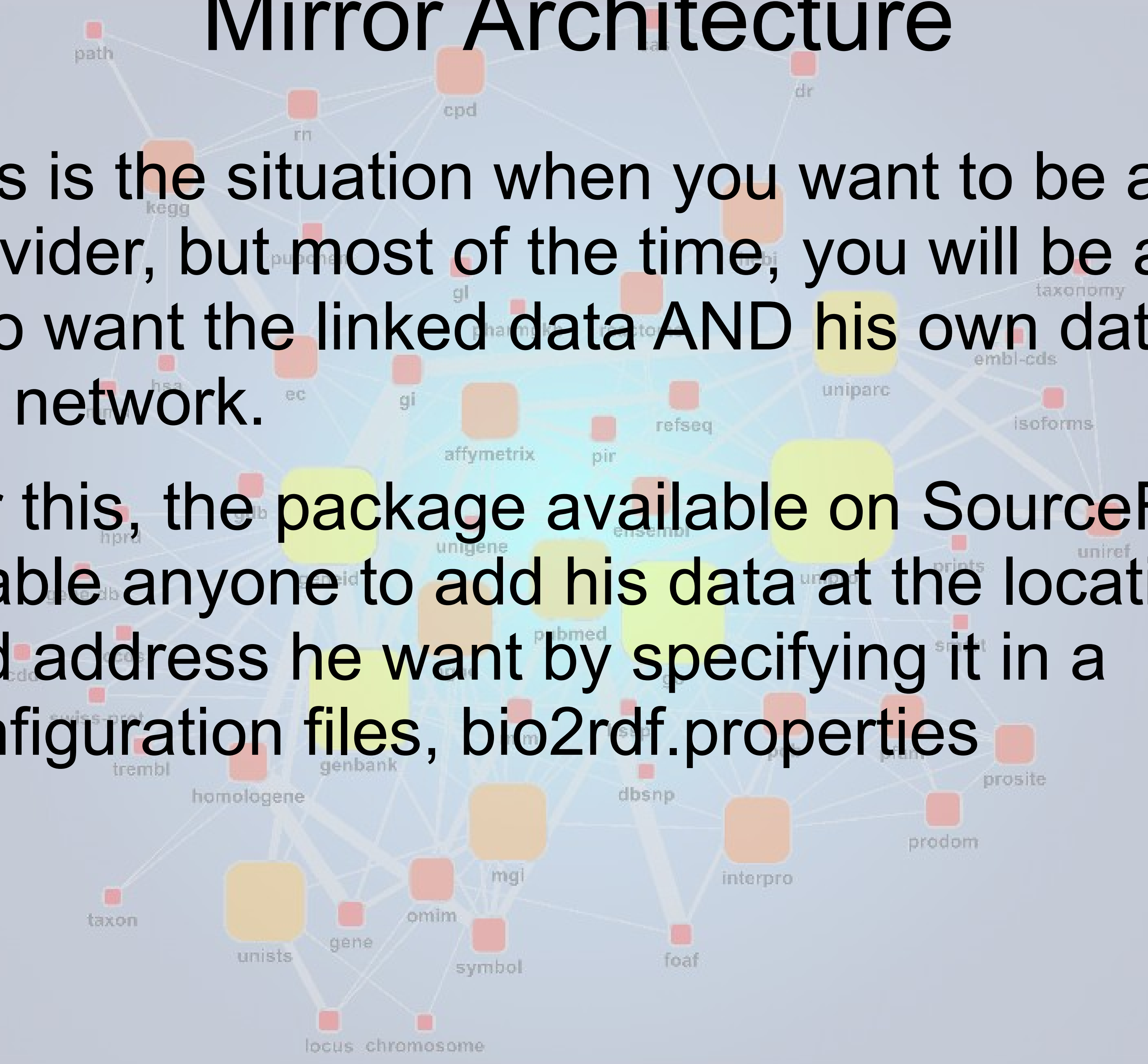
- Use case example for a NCBI Gene ID query


bio2rdf.org
hcls.sciencecommons.org
The RDF document is prepared and sent to the user
user
geneid.domain.name
bio2rdf.org
query = http://domain.name/geneid:15275
DNS Server
dumontierlab.com
DNS Server
http://geneid.domain.name/geneid:15275
sem_web.uq.edu.au
Round robbin in DNS
For this example, 3 domain.name servers mirror exist
domain.name
Round robbin in DNS
Here, 2 geneid.domain.name servers mirror exist

From the URLRewrite file, this rule is triggered

```
<rule>
  <from>^/geneid:15275</from>
  <to type="redirect">http://geneid.bio2rdf.org/geneid:15275</to>
</rule>
```

Mirror Architecture



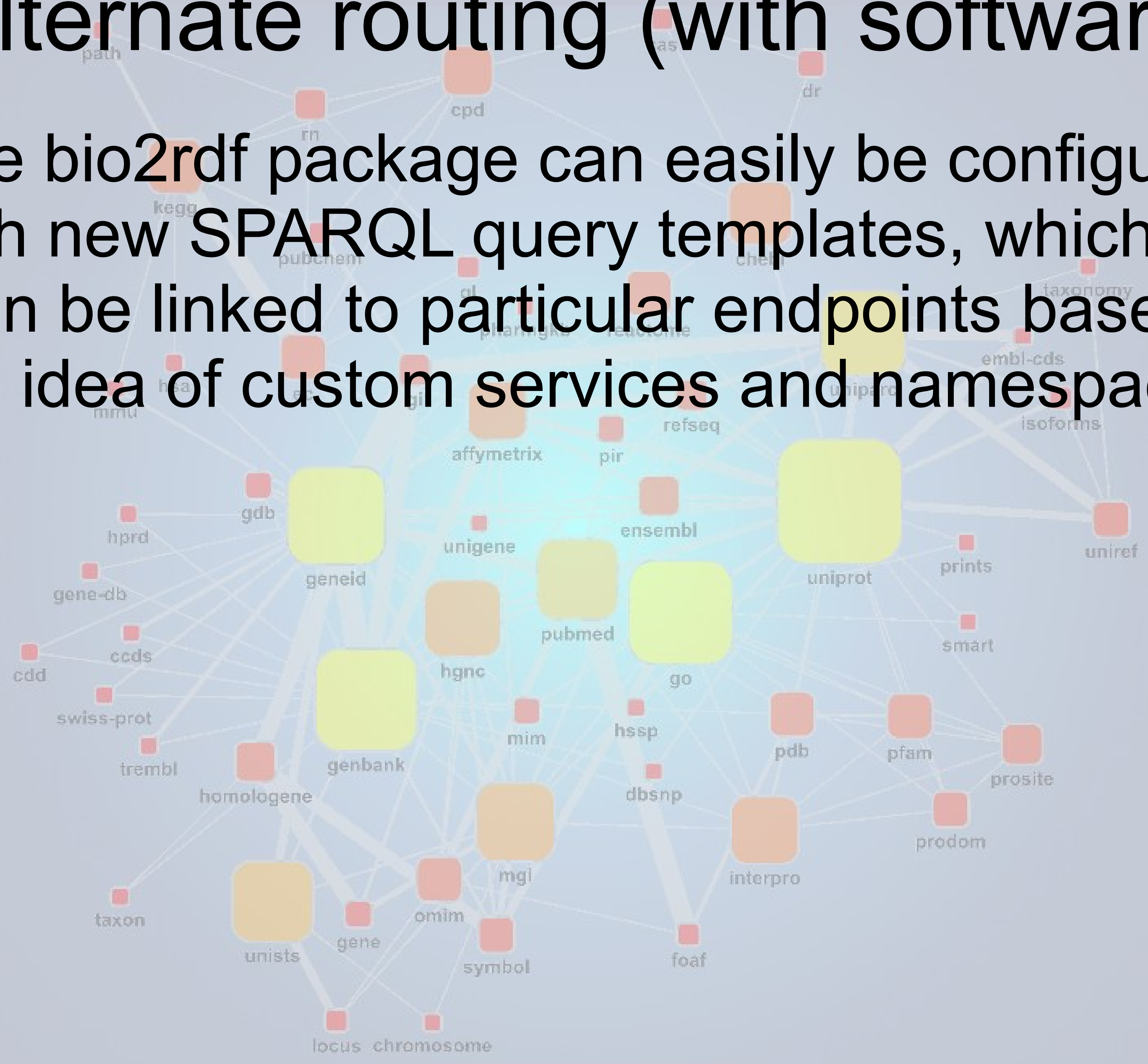
- This is the situation when you want to be a data provider, but most of the time, you will be a user who want the linked data AND his own data in the network.
- For this, the package available on SourceForge enable anyone to add his data at the location and address he want by specifying it in a configuration files, bio2rdf.properties

Alternate routing (with software)

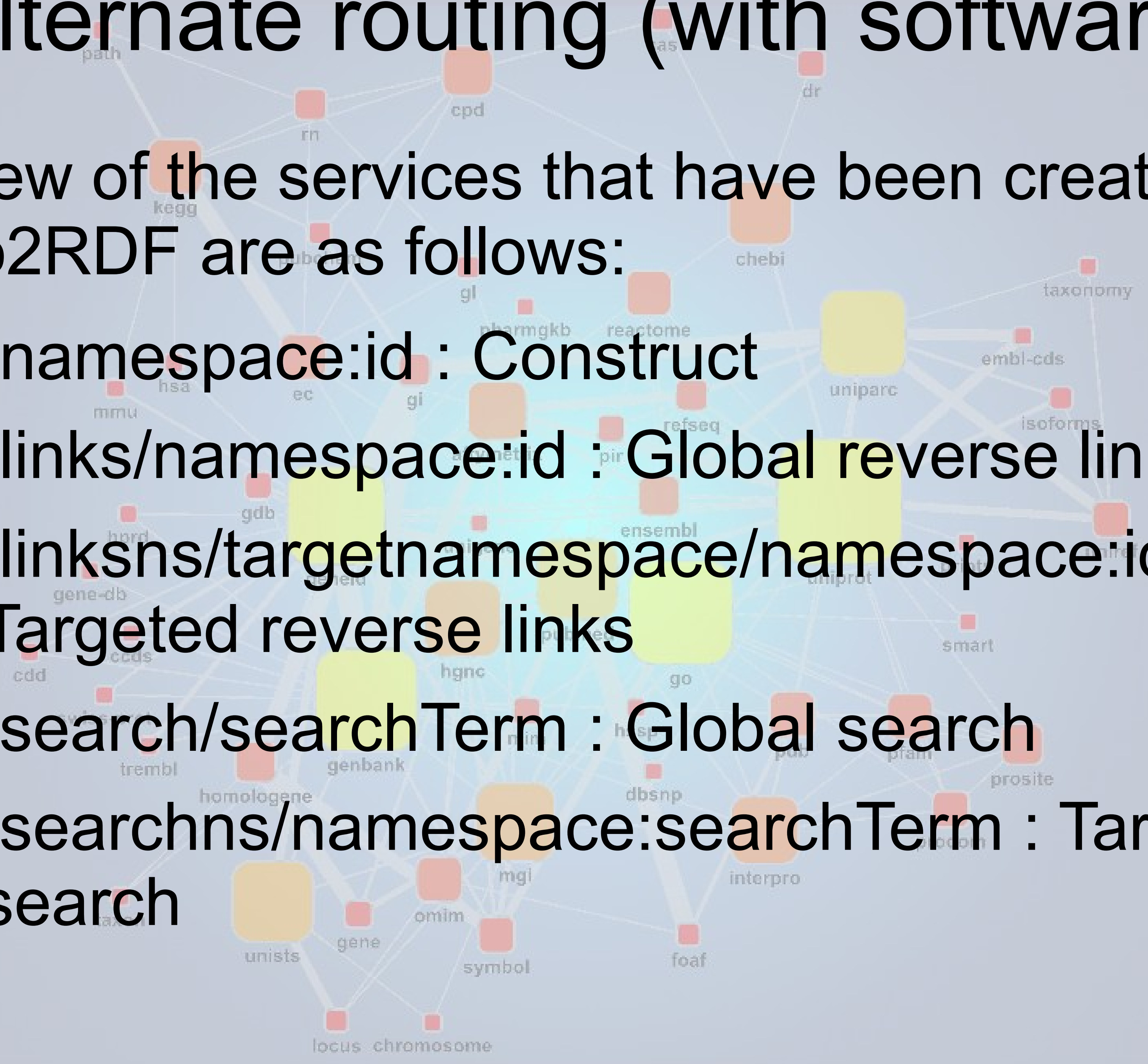
- Although DNS routing is useful for established systems, it is is relatively inflexible, and hard to setup for a private network. A custom software based solution can be used to configure which servers to source data from for particular queries.
- If you are only interested in investigating a small set of databases you can configure the software to only query these servers for your custom services, such as /links/, while retaining the ability to query globally for some services, such as the basic construct query, improving performance without degrading the ability of the server to handle basic Bio2RDF URI's

Alternate routing (with software)

- The bio2rdf package can easily be configured with new SPARQL query templates, which can then be linked to particular endpoints based on the idea of custom services and namespaces.



Alternate routing (with software)



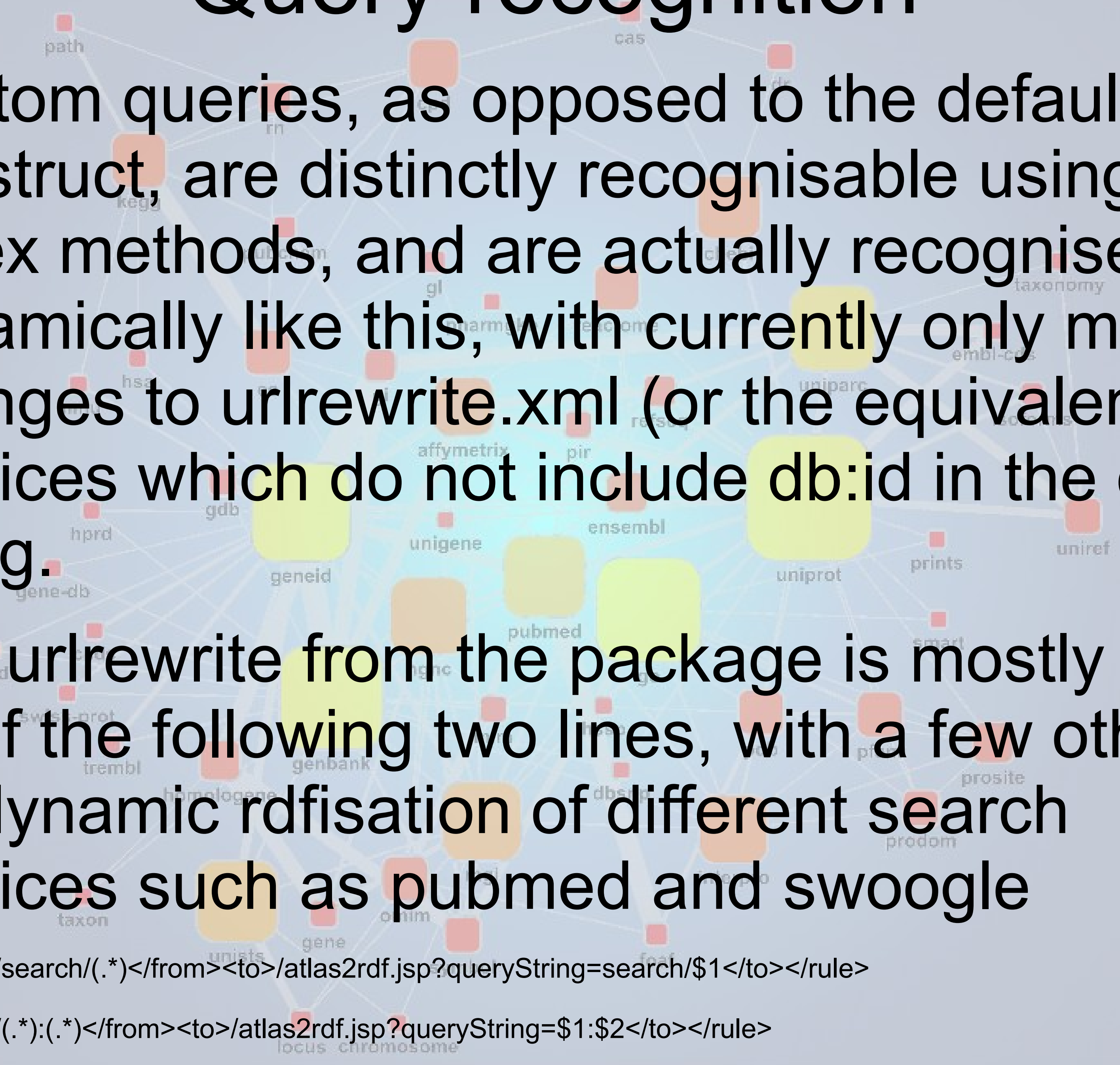
- A few of the services that have been created for Bio2RDF are as follows:
  - /namespace:id : Construct
  - /links/namespace:id : Global reverse links
  - /linksns/targetnamespace/namespace:id : Targeted reverse links
  - /search/searchTerm : Global search
  - /searchns/namespace:searchTerm : Targeted search

Alternate routing (with software)

- Services (continued):
  - /countlinks/namespace:id
  - /constructnodefaults/namespace:id : Construct only on databases which are known to contain a namespace explicitly
  - /constructnoextras/namespace:id : Construct without inserting extra information such as /web/ links
  - /html/namespace:id : Return the URL of an HTML page which contains information about this database record

Alternate routing (with software)

- New services can be created with 11 lines in the bio2rdf.properties configuration file, or less if the service is not a SPARQL query
- These new services can then be assigned to particular providers, which might be SPARQL endpoints or they can be simple URL templates
- Each service can have a different semantic format, eg, /linksns/target/db:id has a different semantic format to /links/db:id, but the standard format /db:id can be derived from both of these using the the STANDARD_URI_TEMPLATE_STRING configuration option

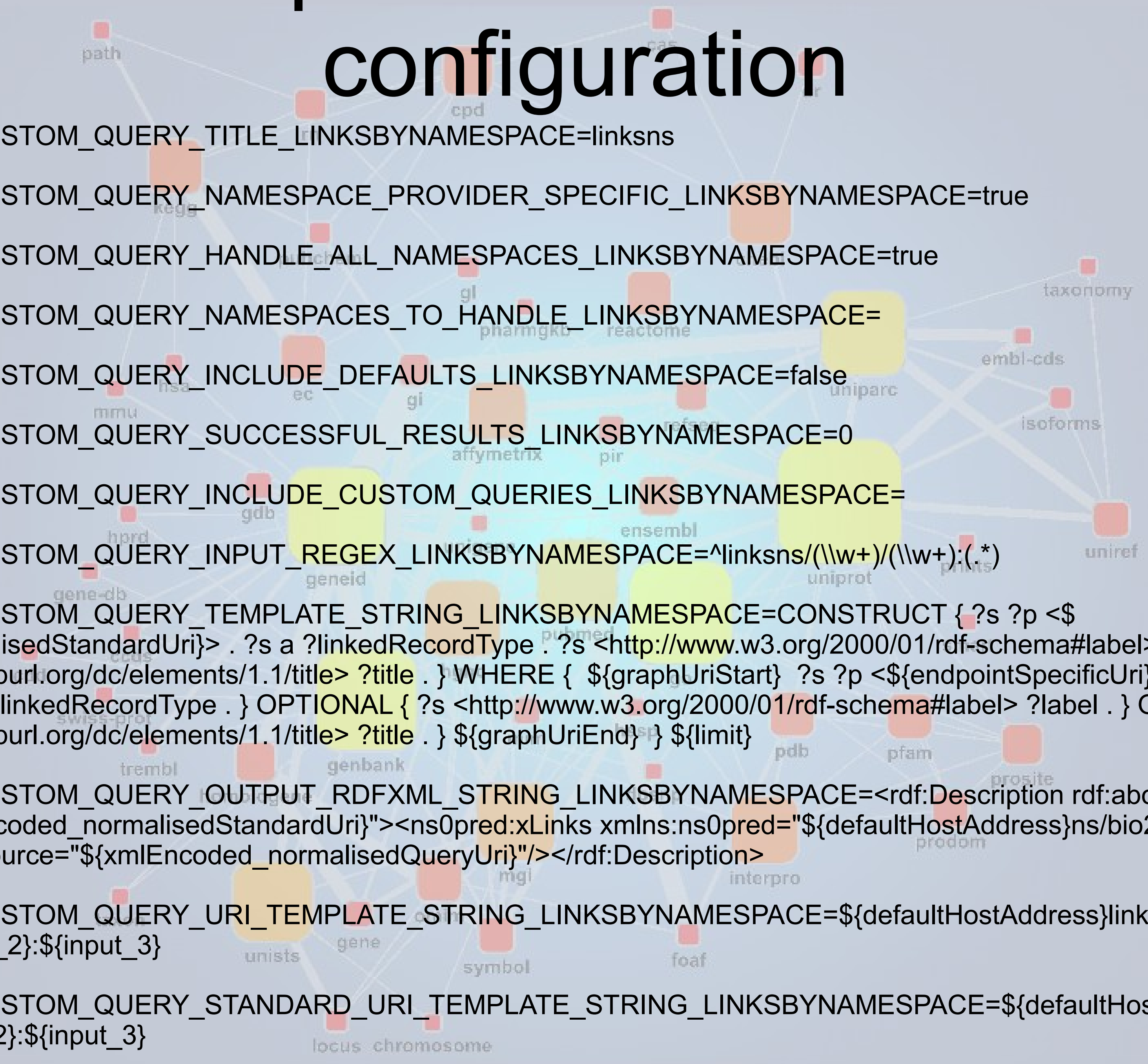


Query recognition

- Custom queries, as opposed to the default construct, are distinctly recognisable using regex methods, and are actually recognised dynamically like this, with currently only minimal changes to urlrewrite.xml (or the equivalent) for services which do not include db:id in the query string.
- The urlrewrite from the package is mostly made up of the following two lines, with a few others for dynamic rdfisation of different search services such as pubmed and swoogle

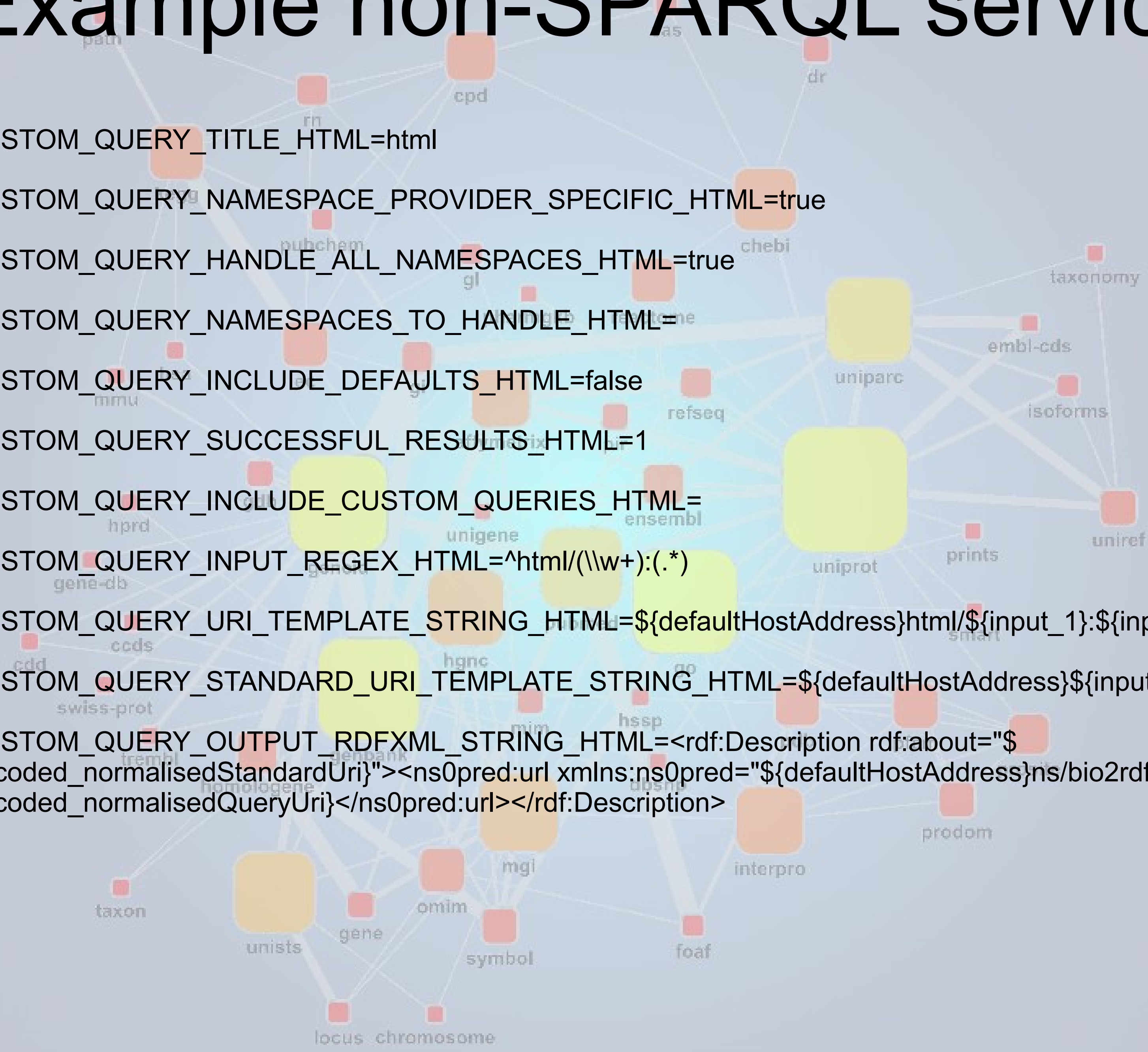
```
<rule><from>^/search/(.*)</from><to>/atlas2rdf.jsp?queryString=search/$1</to></rule>

<rule><from>^/(.*):(.*)</from><to>/atlas2rdf.jsp?queryString=$1:$2</to></rule>
```

Example SPARQL service configuration



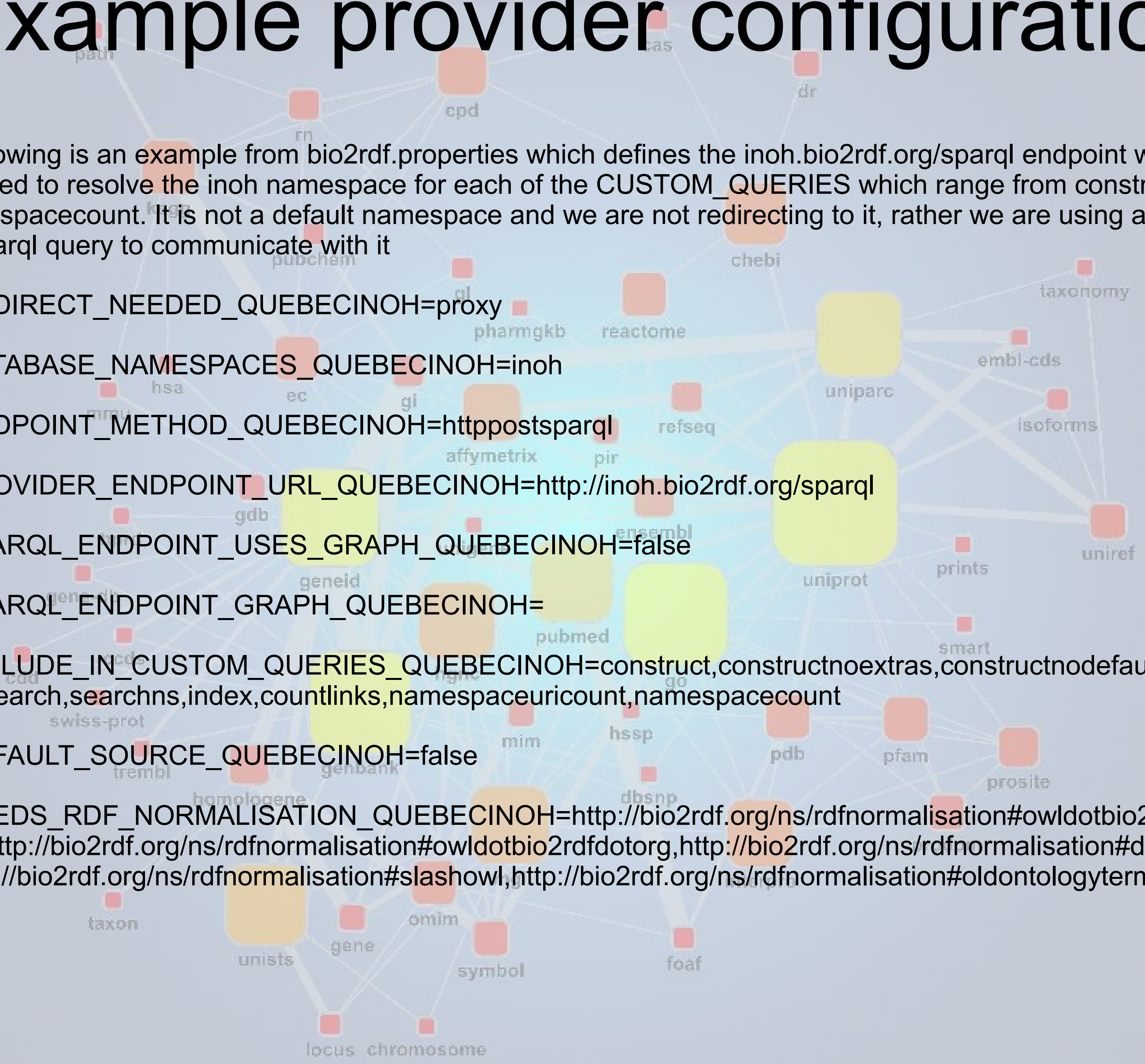
```
Settings.CUSTOM_QUERY_TITLE_LINKSBYNAMESPACE=linksns

Settings.CUSTOM_QUERY_NAMESPACE_PROVIDER_SPECIFIC_LINKSBYNAMESPACE=true

Settings.CUSTOM_QUERY_HANDLE_ALL_NAMESPACES_LINKSBYNAMESPACE=true

Settings.CUSTOM_QUERY_NAMESPACES_TO_HANDLE_LINKSBYNAMESPACE=

Settings.CUSTOM_QUERY_INCLUDE_DEFAULTS_LINKSBYNAMESPACE=false

Settings.CUSTOM_QUERY_SUCCESSFUL_RESULTS_LINKSBYNAMESPACE=0

Settings.CUSTOM_QUERY_INCLUDE_CUSTOM_QUERIES_LINKSBYNAMESPACE=

Settings.CUSTOM_QUERY_INPUT_REGEX_LINKSBYNAMESPACE=^linksns/(\\w+)/(\\w+):(.*)

Settings.CUSTOM_QUERY_TEMPLATE_STRING_LINKSBYNAMESPACE=CONSTRUCT { ?s ?p <${normalisedStandardUri}> . ?s a ?linkedRecordType . ?s <http://www.w3.org/2000/01/rdf-schema#label> ?label . ?s <http://purl.org/dc/elements/1.1/title> ?title . } WHERE {  ${graphUriStart}  ?s ?p <${endpointSpecificUri}> . OPTIONAL { ?s a ?linkedRecordType . } OPTIONAL { ?s <http://www.w3.org/2000/01/rdf-schema#label> ?label . } OPTIONAL { ?s <http://purl.org/dc/elements/1.1/title> ?title . } ${graphUriEnd} } ${limit}

Settings.CUSTOM_QUERY_OUTPUT_RDFXML_STRING_LINKSBYNAMESPACE=<rdf:Description rdf:about="${xmlEncoded_normalisedStandardUri}"><ns0pred:xLinks xmlns:ns0pred="${defaultHostAddress}ns/bio2rdf#" rdf:resource="${xmlEncoded_normalisedQueryUri}"/></rdf:Description>

Settings.CUSTOM_QUERY_URI_TEMPLATE_STRING_LINKSBYNAMESPACE=${defaultHostAddress}linksns/${input_1}/${input_2}:${input_3}

Settings.CUSTOM_QUERY_STANDARD_URI_TEMPLATE_STRING_LINKSBYNAMESPACE=${defaultHostAddress}${input_2}:${input_3}
```

Example non-SPARQL service



```
Settings.CUSTOM_QUERY_TITLE_HTML=html

Settings.CUSTOM_QUERY_NAMESPACE_PROVIDER_SPECIFIC_HTML=true

Settings.CUSTOM_QUERY_HANDLE_ALL_NAMESPACES_HTML=true

Settings.CUSTOM_QUERY_NAMESPACES_TO_HANDLE_HTML=

Settings.CUSTOM_QUERY_INCLUDE_DEFAULTS_HTML=false

Settings.CUSTOM_QUERY_SUCCESSFUL_RESULTS_HTML=1

Settings.CUSTOM_QUERY_INCLUDE_CUSTOM_QUERIES_HTML=

Settings.CUSTOM_QUERY_INPUT_REGEX_HTML=^html/(\\w+):(.*)

Settings.CUSTOM_QUERY_URI_TEMPLATE_STRING_HTML=${defaultHostAddress}html/${input_1}:${input_2}

Settings.CUSTOM_QUERY_STANDARD_URI_TEMPLATE_STRING_HTML=${defaultHostAddress}${input_1}:${input_2}

Settings.CUSTOM_QUERY_OUTPUT_RDFXML_STRING_HTML=<rdf:Description rdf:about="$
    {xmlEncoded_normalisedStandardUri}"><ns0pred:url xmlns:ns0pred="${defaultHostAddress}ns/bio2rdf#">$
    {xmlEncoded_normalisedQueryUri}</ns0pred:url></rdf:Description>
```

Example provider configuration



- The following is an example from bio2rdf.properties which defines the inoh.bio2rdf.org/sparql endpoint which is only configured to resolve the inoh namespace for each of the CUSTOM_QUERIES which range from construct to linksns to namespacecount. It is not a default namespace and we are not redirecting to it, rather we are using a HTTP POST for a sparql query to communicate with it

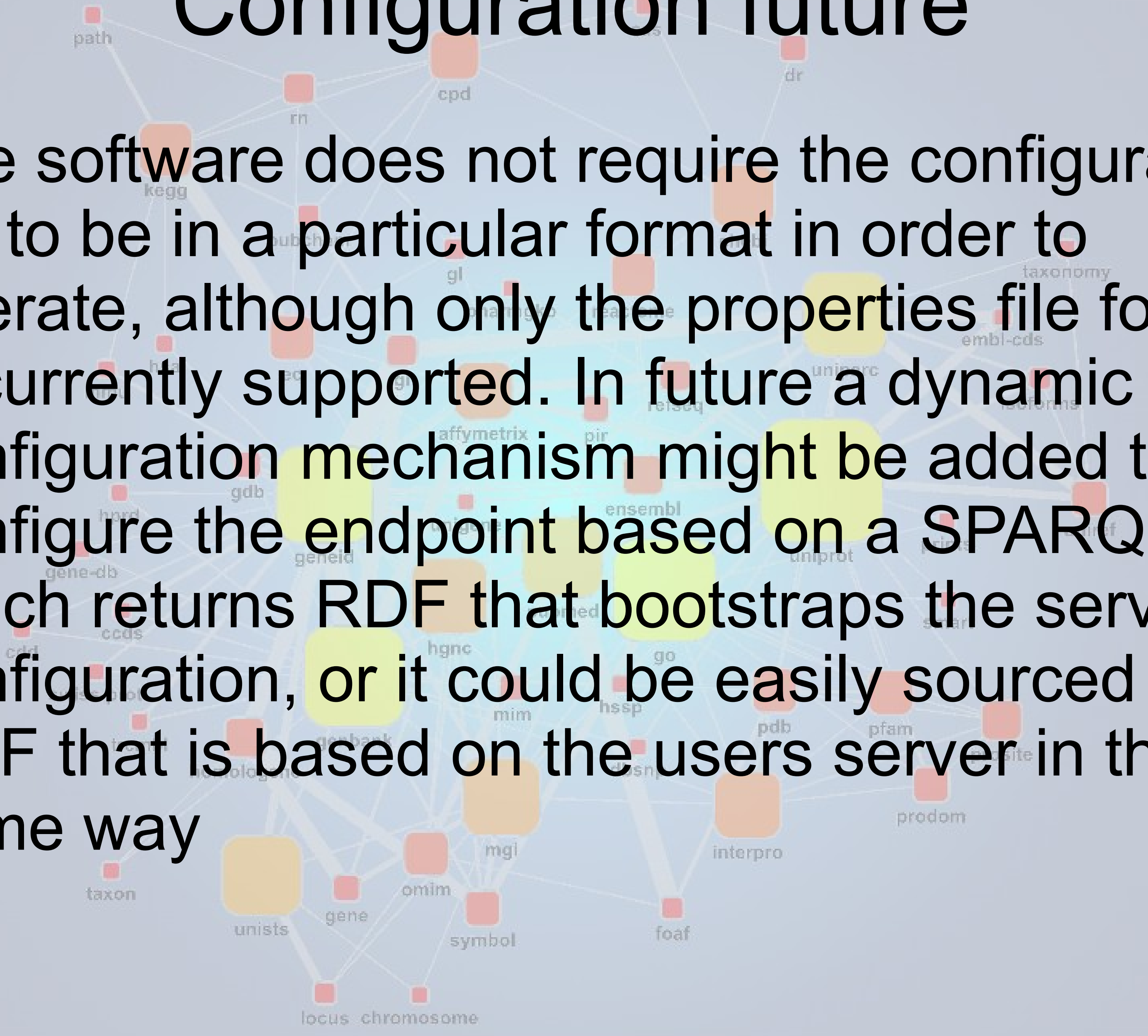
```
Settings.REDIRECT_NEEDED_QUEBECINOH=proxy

Settings.DATABASE_NAMESPACES_QUEBECINOH=inoh

Settings.ENDPOINT_METHOD_QUEBECINOH=httppostsparql

Settings.PROVIDER_ENDPOINT_URL_QUEBECINOH=http://inoh.bio2rdf.org/sparql

Settings.SPARQL_ENDPOINT_USES_GRAPH_QUEBECINOH=false

Settings.SPARQL_ENDPOINT_GRAPH_QUEBECINOH=

Settings.INCLUDE_IN_CUSTOM_QUERIES_QUEBECINOH=construct,constructnoextras,constructnodefaults,label,links,linksns,search,searchns,index,countlinks,namespaceuricount,namespacecount

Settings.DEFAULT_SOURCE_QUEBECINOH=false

Settings.NEEDS_RDF_NORMALISATION_QUEBECINOH=http://bio2rdf.org/ns/rdfnormalisation#owldotbio2rdfdotorgunmangling,http://bio2rdf.org/ns/rdfnormalisation#owldotbio2rdfdotorg,http://bio2rdf.org/ns/rdfnormalisation#dbdotbio2rdfdotorg,http://bio2rdf.org/ns/rdfnormalisation#slashowl,http://bio2rdf.org/ns/rdfnormalisation#oldontologyterms
```

Example non-SPARQL provider

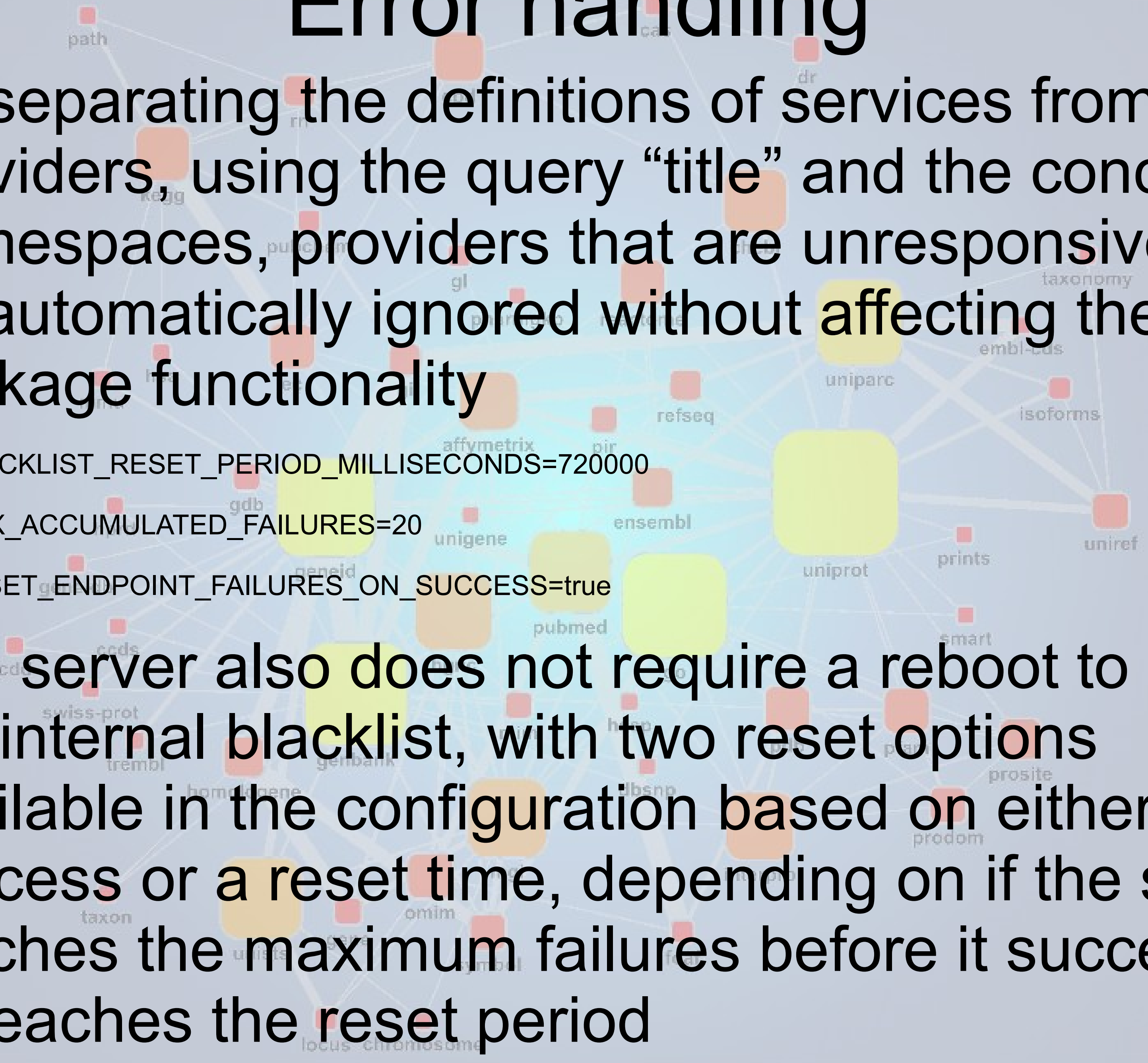
- The following is an example from bio2rdf.properties which defines the providers for geneid for html and xml respectively

```
Settings.ENDPOINT_METHOD_NCBIGENEIDHTML=httpgeturl

Settings.PROVIDER_ENDPOINT_URL_NCBIGENEIDHTML=http://www.ncbi.nlm.nih.gov/sites/entrez?
    cmd=Retrieve&db=gene&dopt=full_report&list_uids=${urlEncoded_input_2}

Settings.DATABASE_NAMESPACES_NCBIGENEIDHTML=geneid

Settings.REDIRECT_NEEDED_NCBIGENEIDHTML=redirect

Settings.INCLUDE_IN_CUSTOM_QUERIES_NCBIGENEIDHTML=html

Settings.DEFAULT_SOURCE_NCBIGENEIDHTML=false


Settings.ENDPOINT_METHOD_NCBIGENEIDXML=httpgeturl

Settings.PROVIDER_ENDPOINT_URL_NCBIGENEIDXML=http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?
    db=gene&retmode=xml&id=${urlEncoded_input_2}

Settings.DATABASE_NAMESPACES_NCBIGENEIDXML=geneid

Settings.REDIRECT_NEEDED_NCBIGENEIDXML=redirect

Settings.INCLUDE_IN_CUSTOM_QUERIES_NCBIGENEIDXML=xml

Settings.DEFAULT_SOURCE_NCBIGENEIDXML=false
```

Configuration future



- The software does not require the configuration file to be in a particular format in order to operate, although only the properties file format is currently supported. In future a dynamic configuration mechanism might be added to configure the endpoint based on a SPARQL call which returns RDF that bootstraps the server configuration, or it could be easily sourced from RDF that is based on the users server in the same way



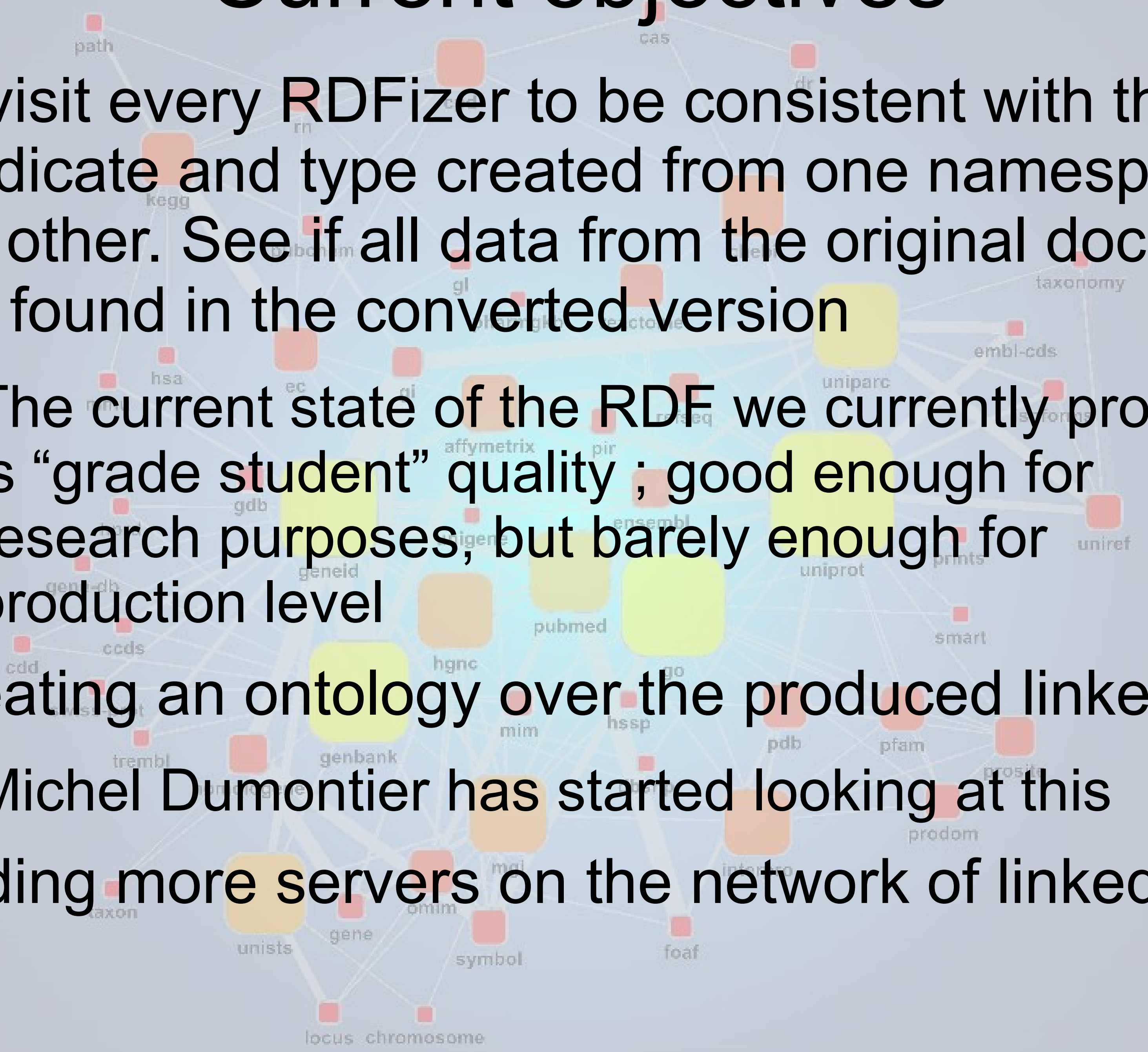
Error handling

- By separating the definitions of services from the providers, using the query “title” and the concept of namespaces, providers that are unresponsive can be automatically ignored without affecting the package functionality

```
Settings.BLACKLIST_RESET_PERIOD_MILLISECONDS=720000

Settings.MAX_ACCUMULATED_FAILURES=20

Settings.RESET_ENDPOINT_FAILURES_ON_SUCCESS=true
```

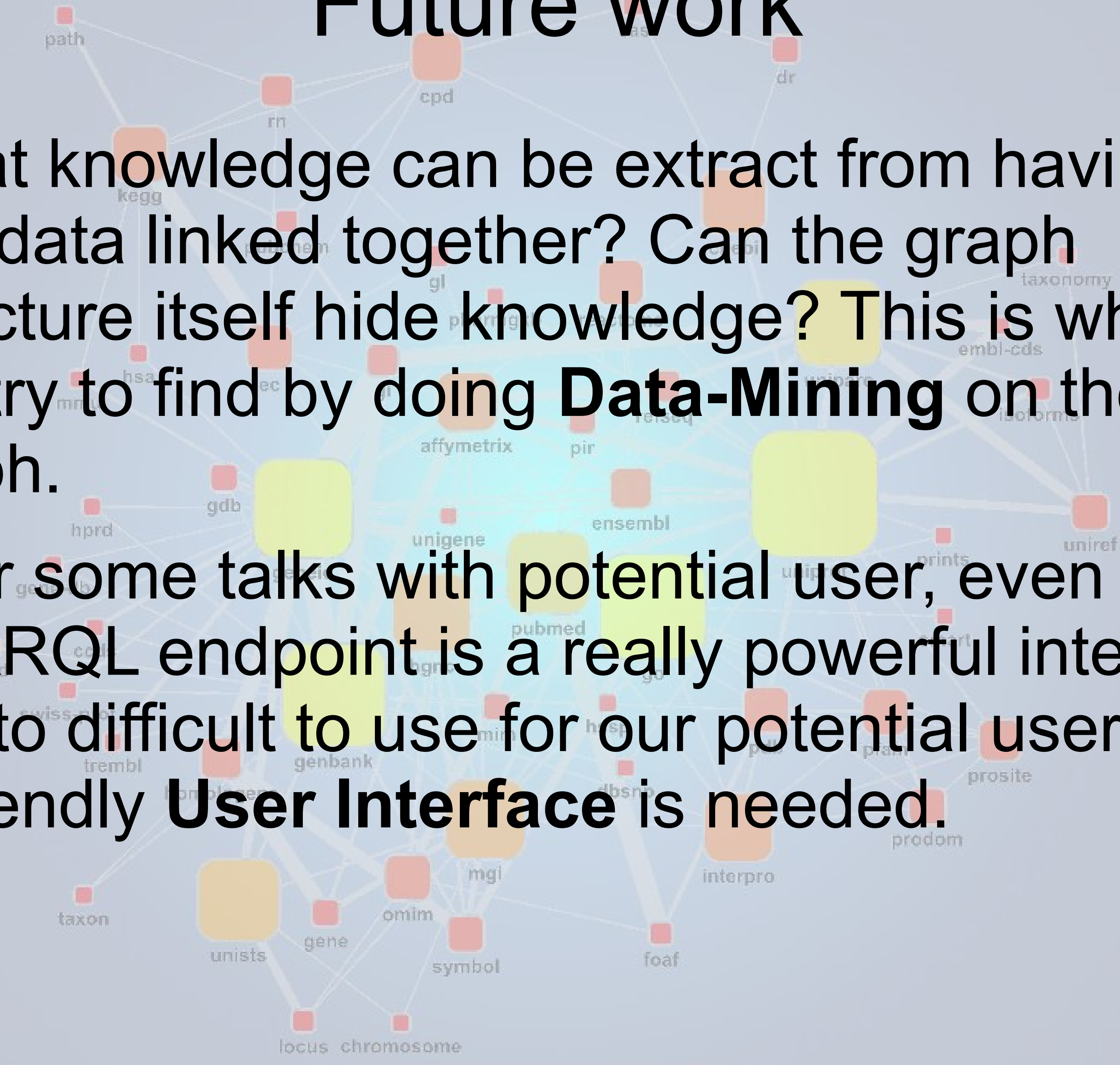
- The server also does not require a reboot to reset the internal blacklist, with two reset options available in the configuration based on either success or a reset time, depending on if the server reaches the maximum failures before it succeeds or reaches the reset period

Current objectives



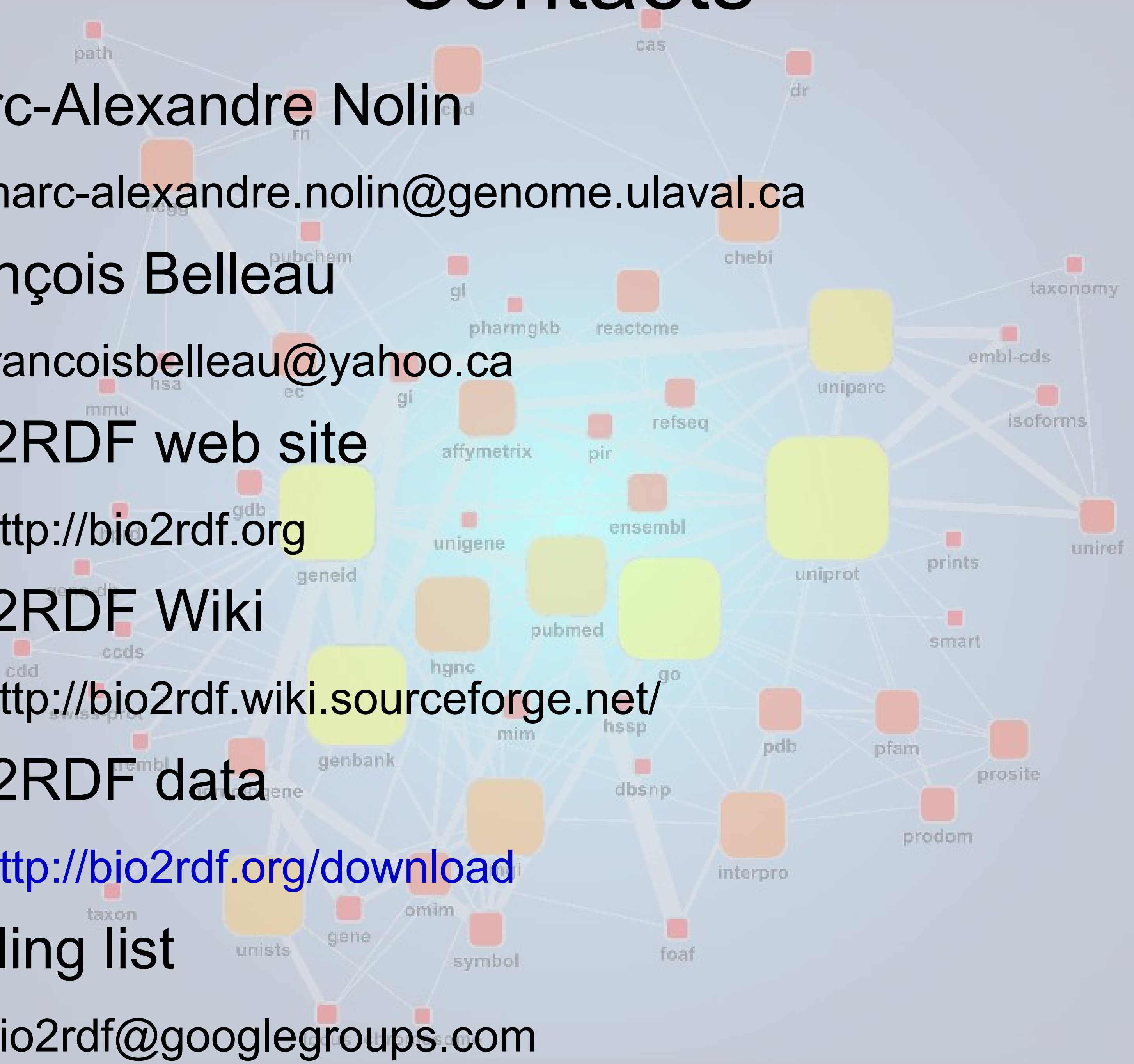
- Revisit every RDFizer to be consistent with the predicate and type created from one namespace to the other. See if all data from the original document are found in the converted version
  - The current state of the RDF we currently produce is “grade student” quality ; good enough for research purposes, but barely enough for production level
- Creating an ontology over the produced linked data
  - Michel Dumontier has started looking at this
- Adding more servers on the network of linked data

Future work



- What knowledge can be extract from having all this data linked together? Can the graph structure itself hide knowledge? This is what we will try to find by doing **Data-Mining** on the graph.
- After some talks with potential user, even if SPARQL endpoint is a really powerful interface, it is to difficult to use for our potential users. So a friendly **User Interface** is needed.

Contacts



- Marc-Alexandre Nolin
  - marc-alexandre.nolin@genome.ulaval.ca
- François Belleau
  - francoisbelleau@yahoo.ca
- Bio2RDF web site
  - http://bio2rdf.org
- Bio2RDF Wiki
  - http://bio2rdf.wiki.sourceforge.net/
- Bio2RDF data
  - http://bio2rdf.org/download
- Mailing list
  - bio2rdf@googlegroups.com

Thanks

- Peter Ansell, François Belleau, Kingsley Idehen, Michel Dumontier, Nicole Tourigny, Paul Roe, James M Hogan, Philippe Rigault
- The Bio2RDF community
- Arborea Project
- Centre de recherche du CHUL
- Université Laval
- QUT eResearch Center
- Openlink Virtuoso