INTRIG
INFORMATION & NETWORKING TECHNOLOGIES RESEARCH & INNOVATION GROUP

Aplicação de Bancos de Dados Baseados em Grafos no Controle de Redes de Computadores

Canditada: Talita de Paula C. de Souza

Orientador: Prof. Dr. Christian Esteve Rothenberg

Coorientador: Prof. Dr. Luciano Bernardes de Paula

FEEC
Faculdade de Engenharia Elétrica e de Computação
UNICAMP

29 de Abril de 2016

Agenda

- Introdução
- Objetivos
- Fundamentação Teórica
  - Redes Definidas por Software
  - Virtualização de Funções de Rede
  - Modelos Semânticos
  - Bancos de Dados Baseados em Grafos
- Casos de Uso
  - Modelagem Semântica
  - Multidomínios SDN
  - Virtualização Recursiva
- Conclusão e Trabalhos Futuros

Agenda

- Introdução
- Objetivos
- Fundamentação Teórica
  - Redes Definidas por Software
  - Virtualização de Funções de Rede
  - Modelos Semânticos
  - Bancos de Dados Baseados em Grafos
- Casos de Uso
  - Modelagem Semântica
  - Multidomínios SDN
  - Virtualização Recursiva
- Conclusão e Trabalhos Futuros

Introdução

- ✓ Crescimento de soluções em Redes Definidas por Software e Funções de Rede Virtualizadas;
- ✓ Representação detalhada e a manutenção de modelos de informação sobre sua topologia necessárias em gerência de redes;
- ✓ Crescimento da utilização de metadados compatíveis com os padrões da Web Semântica;
- ✓ Popularidade e crescimento de banco de dados NoSQL, especialmente baseado em grafos;

Agenda

- Introdução
- Objetivos
- Fundamentação Teórica
  - Redes Definidas por Software
  - Virtualização de Funções de Rede
  - Modelos Semânticos
  - Bancos de Dados Baseados em Grafos
- Casos de Uso
  - Modelagem Semântica
  - Multidomínios SDN
  - Virtualização Recursiva
- Conclusão e Trabalhos Futuros

Objetivos

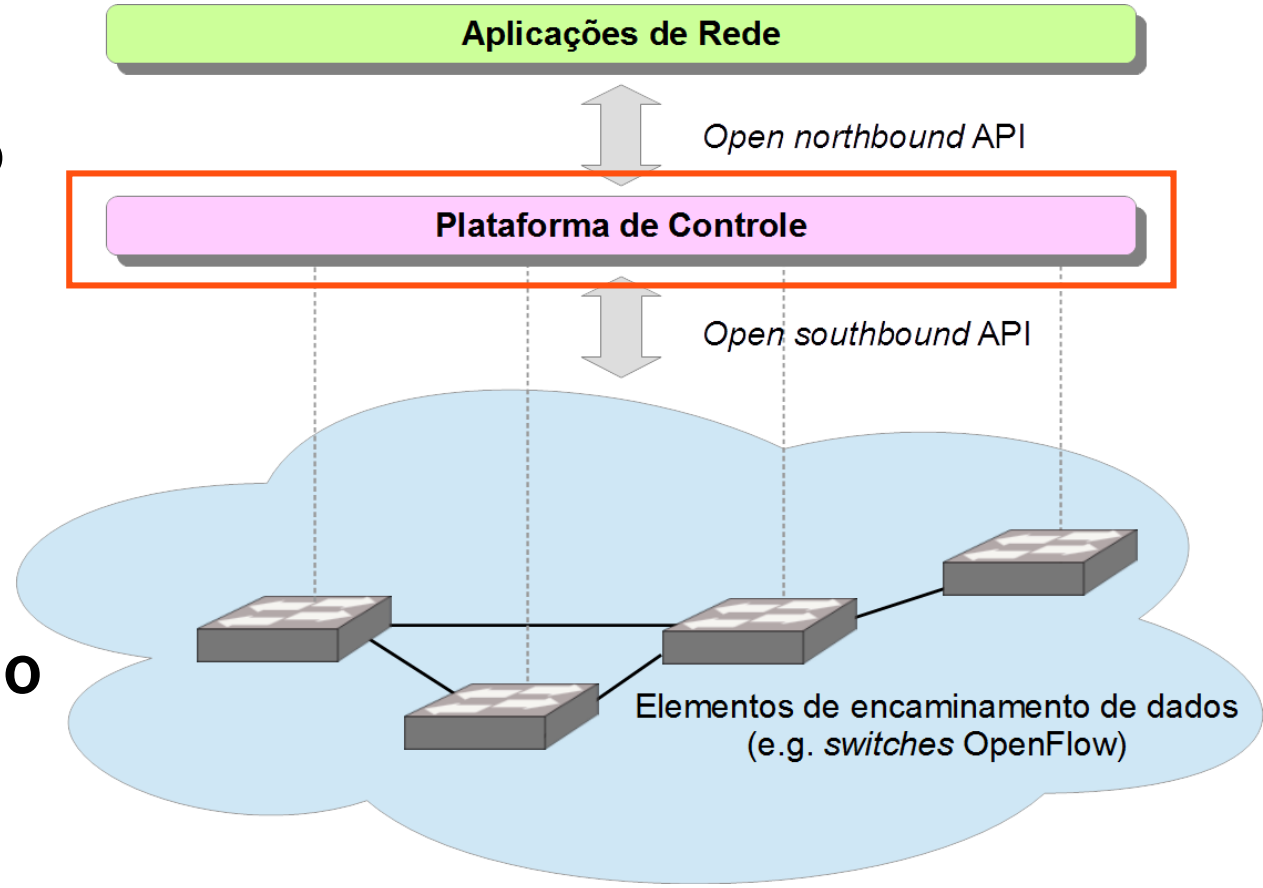
- Aplicação de bancos de dados baseados em grafos no contexto de controle de redes de computadores;
  - Utilização de modelos semânticos;
  - Suporte de abstração de rede para controladores e/ou orquestradores;
  - Interoperabilidade entre controladores;

Agenda

- Introdução
- Objetivos
- **Fundamentação Teórica**
  - **Redes Definidas por Software**
  - **Virtualização de Funções de Rede**
  - **Modelos Semânticos**
  - **Bancos de Dados Baseados em Grafos**
- Casos de Uso
  - Modelagem Semântica
  - Multidomínios SDN
  - Virtualização Recursiva
- Conclusão e Trabalhos Futuros

Redes Definidas por Software

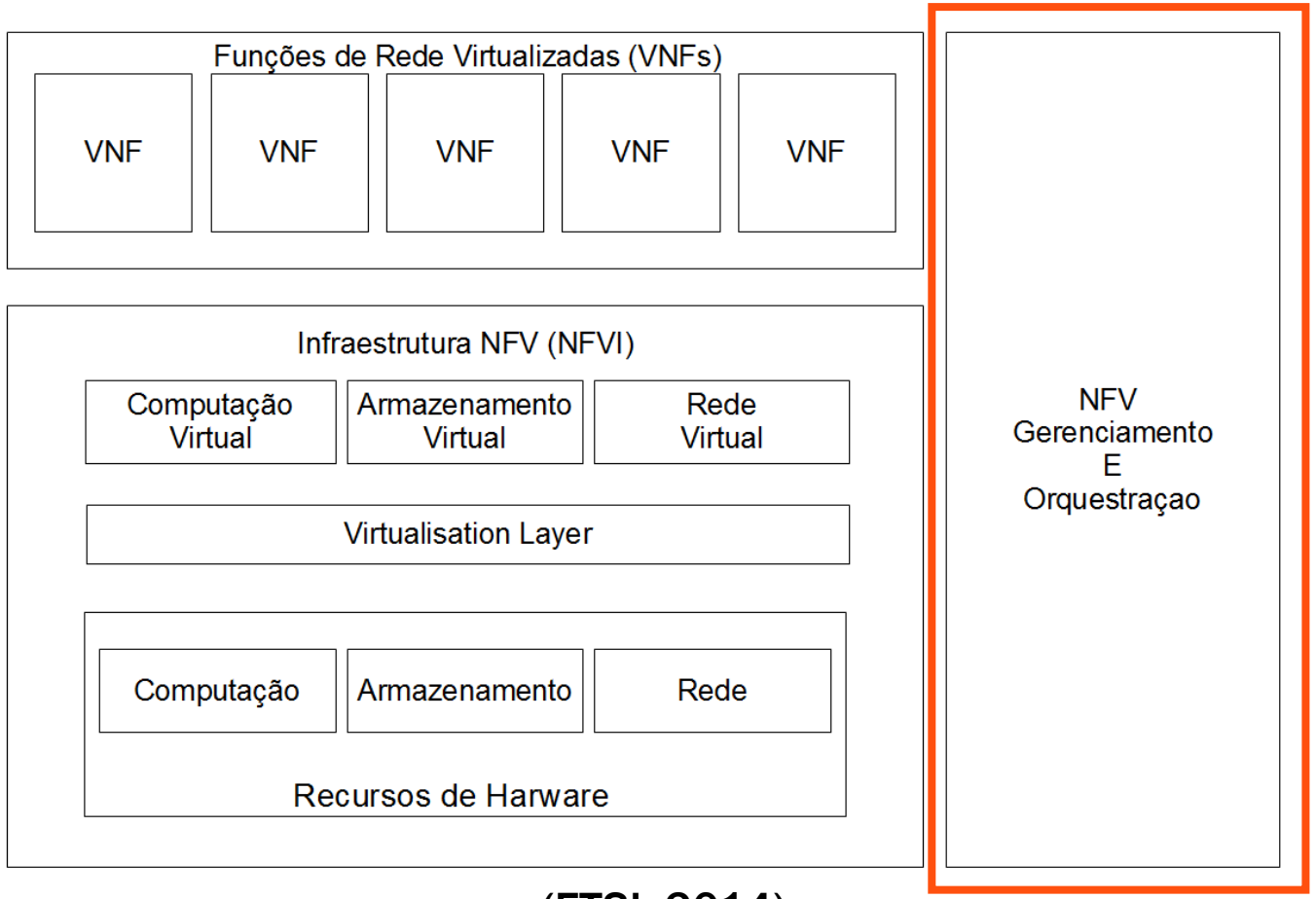
- ✓*Software Defined Network* (SDN):
  - ✓Plano de Controle desacoplado do Plano de Encaminhamento;
  - ✓Novas abstrações de controle e encaminhamento das redes;
- ✓API programática para a abstração de fluxos de pacotes:
  - ✓Protocolo *OpenFlow*;
- ✓Abstração de topologia de rede;



KREUTZ, D. et al., 2015

Virtualização de Funções de Rede

- ✓*Network Function Virtualization* (NFV):
  - ✓Provisionamento de serviços de telecomunicações;
- ✓Desacoplamento de *Software* e *Hardware;*
- ✓Desenvolvimento flexível de função de rede;
- ✓Escala dinâmica;

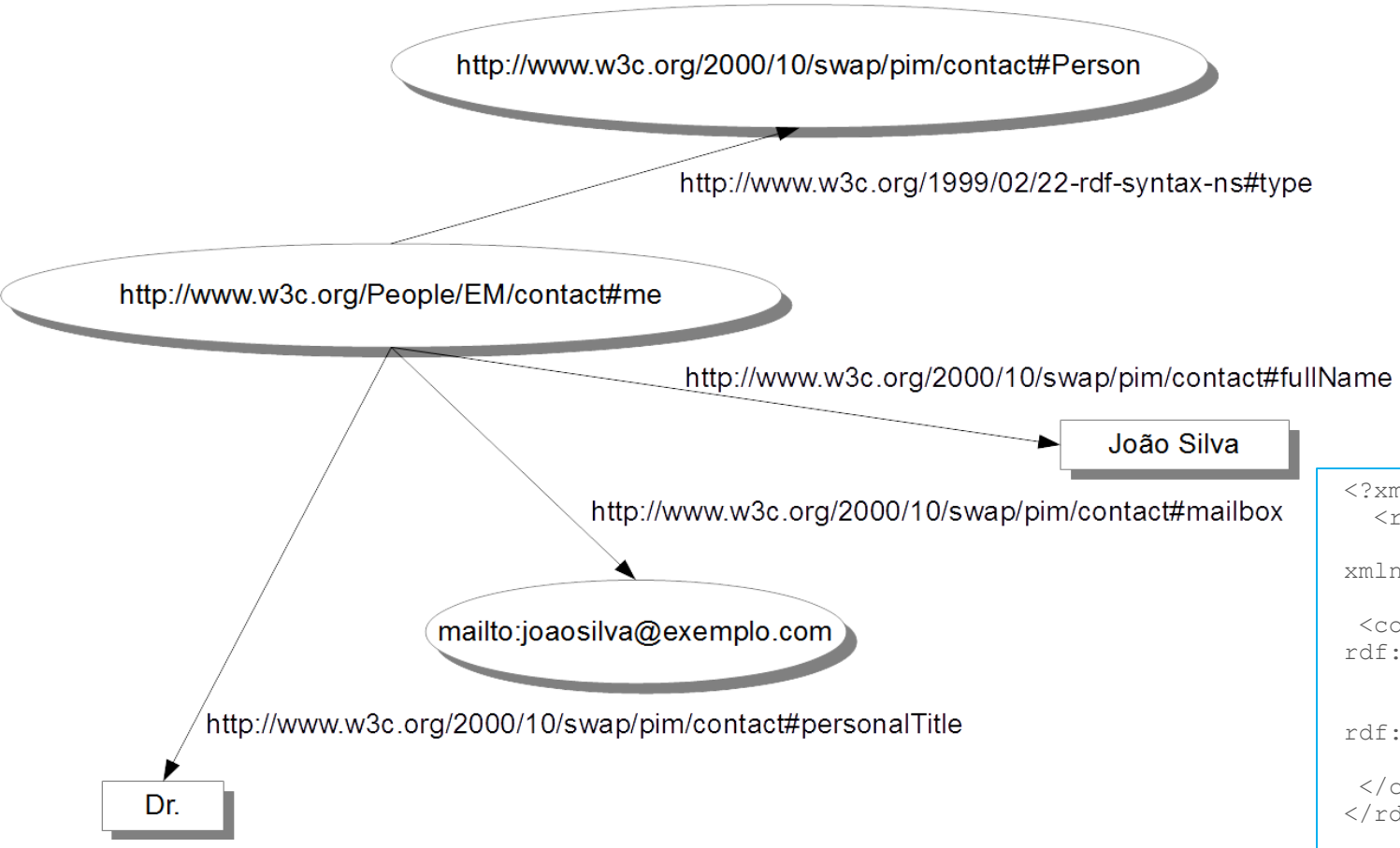


(ETSI, 2014)

Modelos Semânticos

- ✓ Web Semântica:
  - ✓ Permite reutilização da informação;
  - ✓ Integração de dados entre diversos órgãos e instituições;
  - ✓ Buscas aprimoradas na Web;
  - ✓ Garantia de acessibilidade;
  - ✓ Interligar recursos por meio de *Universal Resource Identifiers* (URIs);
    - ✓ Objetos e Relacionamentos;
  - ✓ *Resource Description Framework* (RDF):
    - ✓ sujeito → predicado → objeto
  - ✓ *Web Ontology Language* (OWL);
    - ✓ *Class, Individual, Object Property, Data Property*

Modelos Semânticos




```
<?xml version="1.0"?>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#" >

 <contact:Person
rdf:about="http://www.w3.org/People/EM/contact#me">
           <contact:fullName>João Silva</contact:fullName>
           <contact:mailbox
rdf:resource="mailto:joao@exemplo.org"/>
           <contact:personalTitle>Dr.</contact:personalTitle>
 </contact:Person>
</rdf:RDF>
```

Modelos Semânticos

- *Network Markup Language* – NML (van der Ham et al., 2013)
  - Descreve redes multicamadas e multidomínios:
    - Rede Virtualizada;
    - Rede Utilizando Diferentes Tecnologias;
  - *Network Markup Language Working Group* (NML-WG) no *Open Grid Forum* (OGF)

Modelos Semânticos

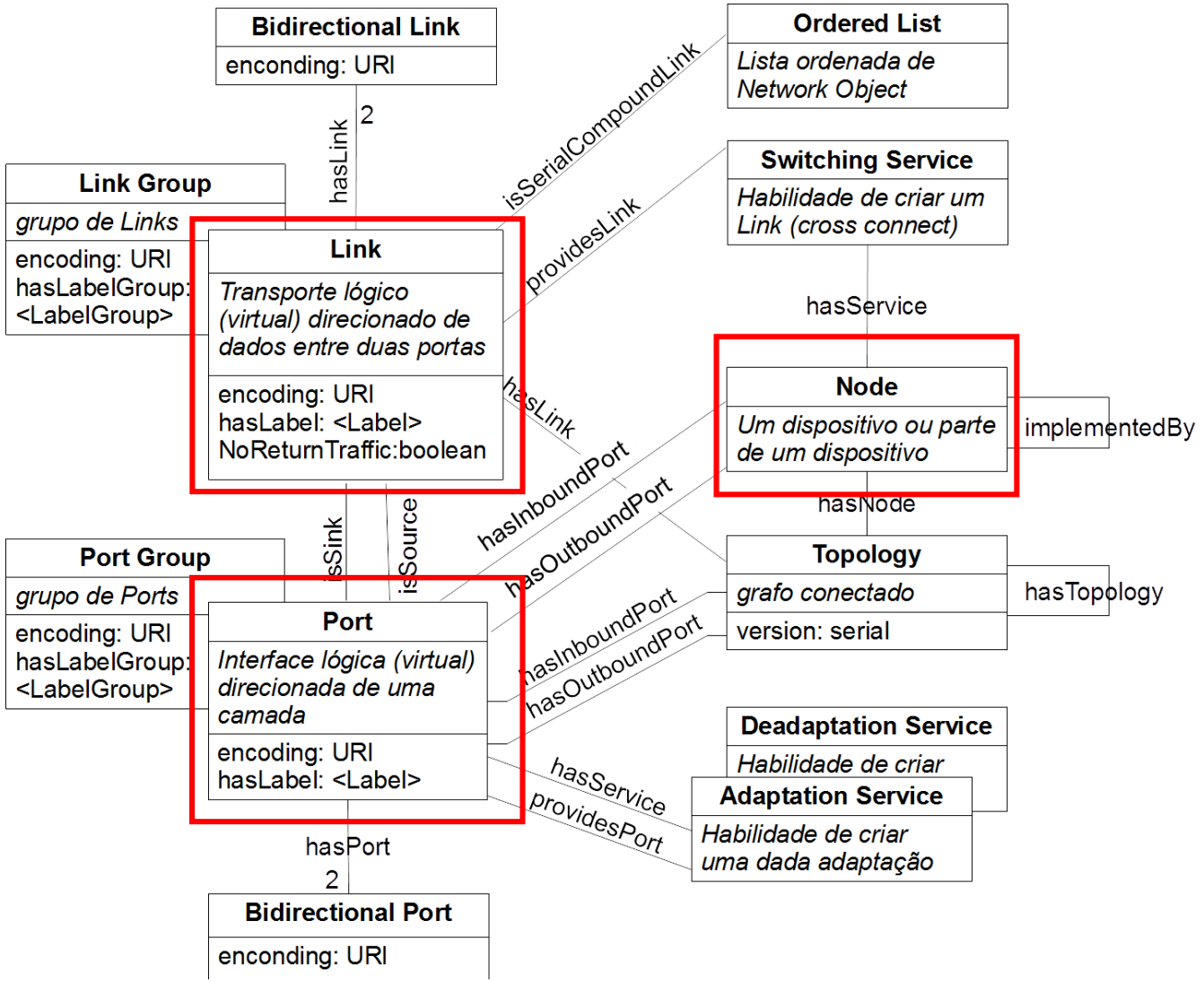
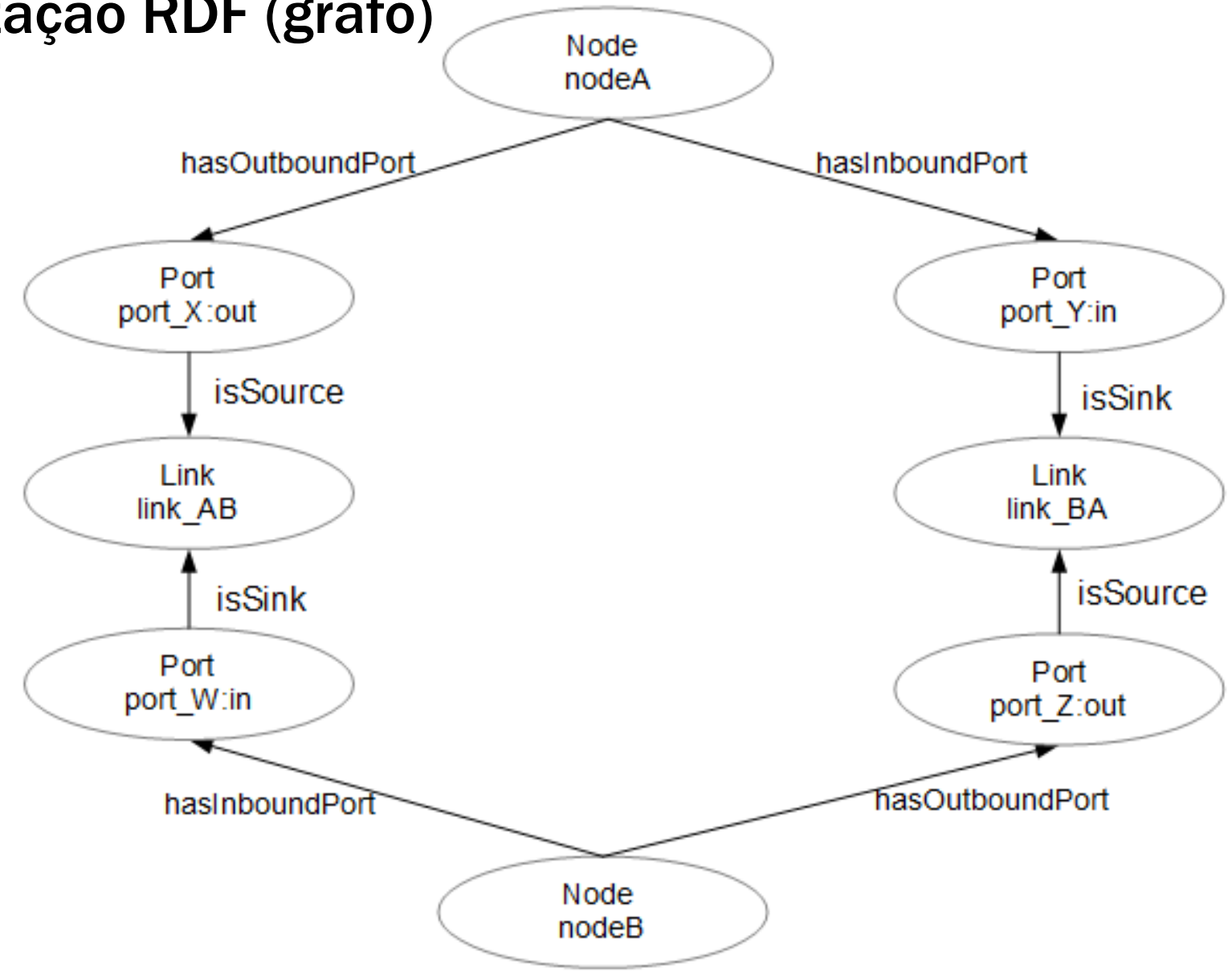


Diagrama de Classes UML – Schema NML

Modelos Semânticos
Exemplo NML

- Representação RDF (grafo)



Modelos Semânticos

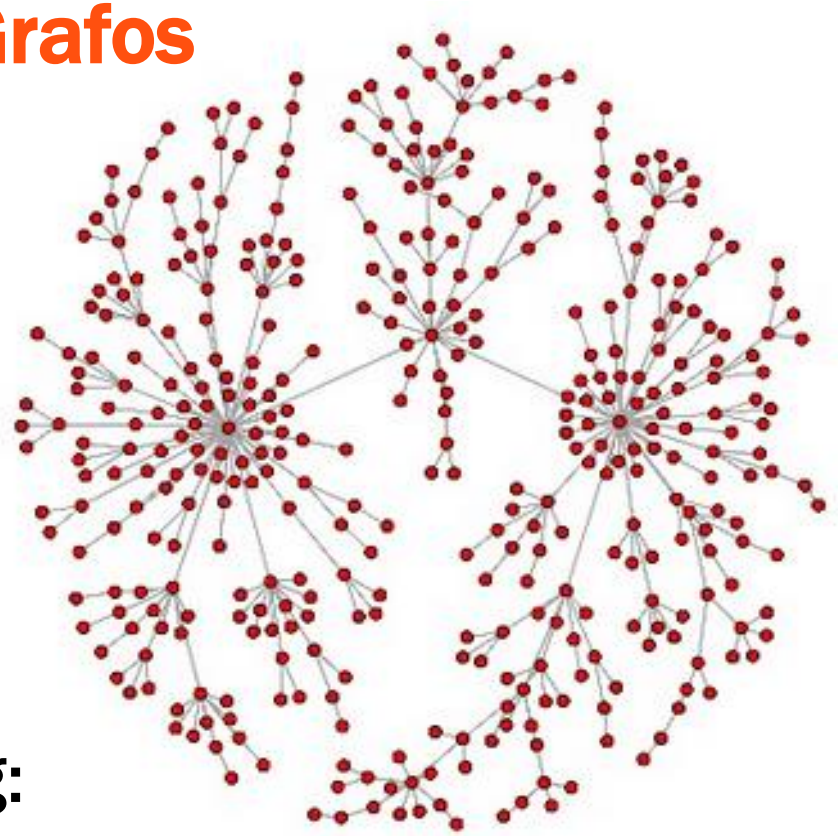
- Extensões do *Network Markup Language* – NML (van der Ham et al., 2013) conforme a necessidade:
- *Infrastructure and Network Description Language* (INDL);
- Projetos:
  - NOVI – Plataformas para Internet do Futuro:
    - http://www.fp7-novi.eu/
  - GEYSERS – Virtualização de Redes Ópticas:
    - http://www.i2cat.net/en/projects/geysers
  - CINEGRID – Distribuição de Cinema Digital:
    - http://www.cinegrid.org

Armazenamento de Dados

- ✓ **Banco de Dados Relacional**
  - ✓ Consolidado; Bem Documentado;
  - ✓ Transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade)
  - ✓ Limitações:
    - ✓ Consultas em dados altamente conectados;
    - ✓ Modelagem dos dados de forma adaptada;
- ✓ **NOSQL** (http://nosql-database.org/)
  - ✓ Livre de esquema;
  - ✓ Escalabilidade;
  - ✓ Disponibilidade;
  - ✓ Menor tempo de resposta;
  - ✓ Escalonamento Horizontal;

Armazenamento de Dados
Bancos de Dados Baseados em Grafos

- ✓Grafo:
  - ✓Vértices;
  - ✓Arestas;
- ✓Topologia;
- ✓Interconectividade de Dados (Foco no relacionamento);
- ✓Modelagem natural de problemas, e.g:
  - ✓Web semântica;
  - ✓Redes de computadores;
  - ✓Motores de recomendação, etc;

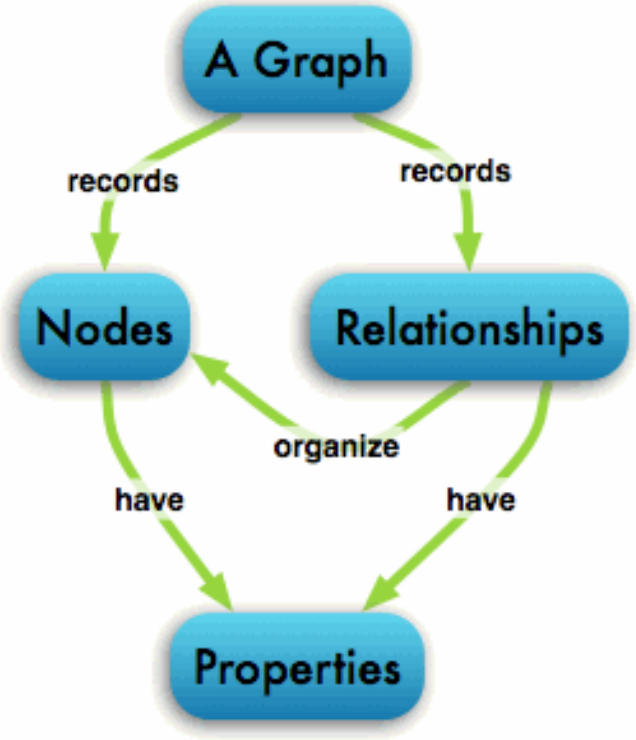


Armazenamento de Dados
Bancos de Dados Baseados em Grafos

- Comparação de GBDs a partir de Benchmark (Jouili e Vansteenberghe, 2013):
  - Neo4j, OrientDB, Titan e DEX;
  - Para consultas o que obteve melhores resultados foi o Neo4j;
- Comparação de Linguagens de Consulta no Neo4j (Holzschuher; Peinl, 2013):
  - Cypher, Gremlin, API Java;
  - Análise de desempenho, compreensibilidade e linhas de código.
- Auditoria em Arquitetura Virtualizada (*Cloud)* (Soundararajan and Kakaraddi, 2014):
  - Neo4j e *Cypher*:
    - Análise de Risco;
    - Reporte Simples;
    - Comparação de Inventário;

Armazenamento de Dados
Neo4j

- Armazenamento Nativo de Grafo e Processamento Nativo de Grafo;
- Open Source (Versão *Community*);
- Modelo Grafo de Propriedade (Property Graph):
  - Nós e relacionamentos possuem propriedades;
  - Nós, relacionamentos e *labels*;
- Linguagem de Consulta:
  - *Cypher;*
  - Gremlin (TinkerPop);
  - API Java.



Armazenamento de Dados
Neo4j

- **Cypher**
  - **Linguagem de consulta SQL *like;***

```
MATCH (n:Node)-[:hasOutboundPort]->(p:Port)-[:isSource]->(l:Link)
WHERE n.name="A"
RETURN COUNT(l) AS CountOutDegree
```

```
MATCH (a:Port), (b:Link)
WHERE a.name="A_out" AND b.name="A_B"
CREATE (a)-[r:isSource]->(b)
RETURN r
```

Contribuições Científicas

- Notação semântica (NML) no contexto de controladores SDN com interfaces a GDBs (Neo4j) e avaliação experimental;
- Mapeamento de primitivas de uma aplicação SDN em consultas utilizando API do banco de dados;
- Identificação de limitações do modelo NML no suporte de primitivas de aplicação controle SDN;
- Formalização do *parsing* do modelo semântico para o banco de dados e vice-versa.
- Indexação de dados nos cenários de multidomínios SDN e virtualização recursiva de NFV;
- Estudo de extensão do modelo semântico para suporte de tabelas de roteamento.

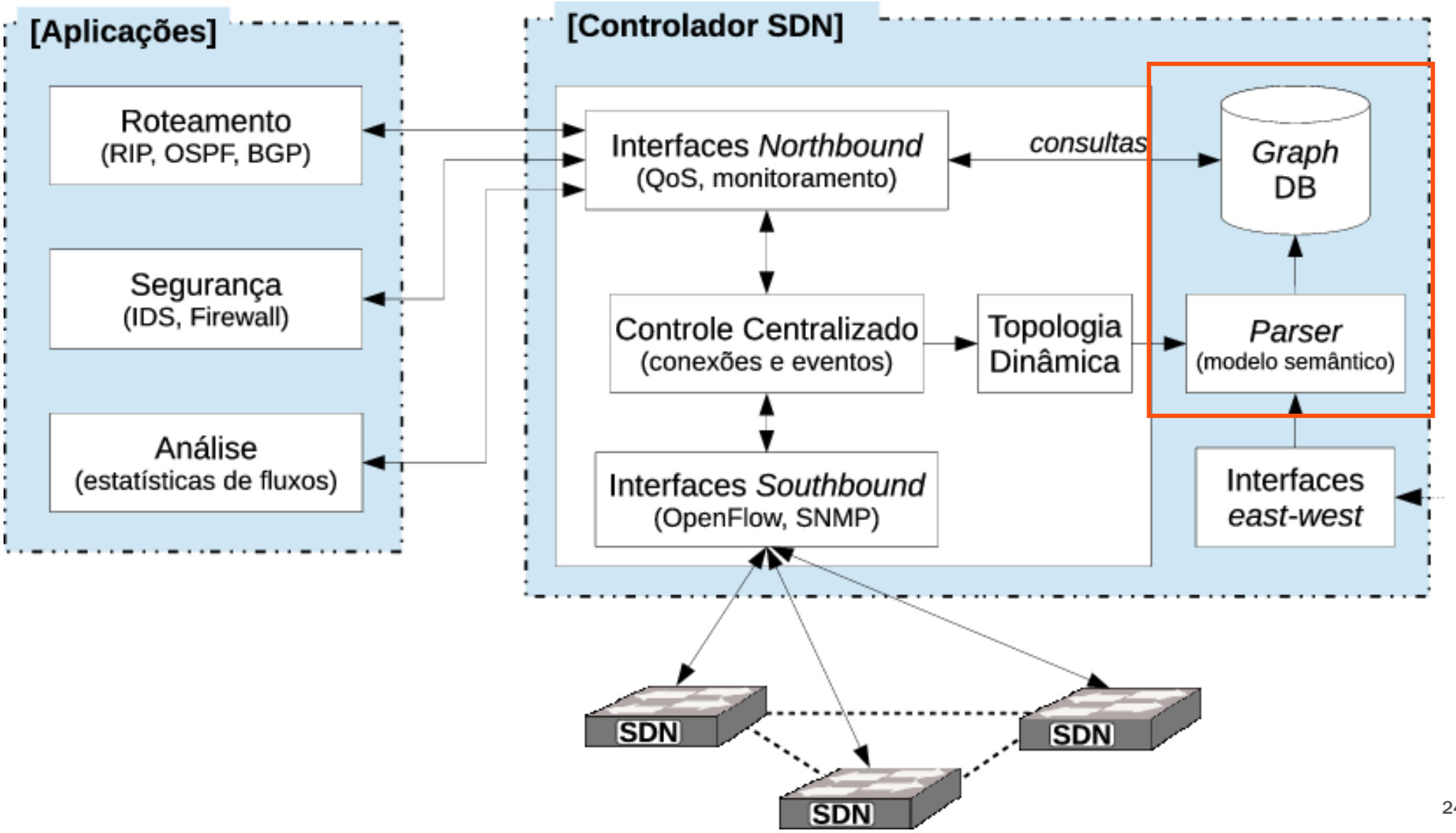
Agenda

- Introdução
- Objetivos
- Fundamentação Teórica
  - Redes Definidas por Software
  - Virtualização de Funções de Rede
  - Modelos Semânticos
  - Bancos de Dados Baseados em Grafos
- **Casos de Uso**
  - **Modelagem Semântica**
  - **Multidomínios SDN**
  - **Virtualização Recursiva**
- Conclusão e Trabalhos Futuros

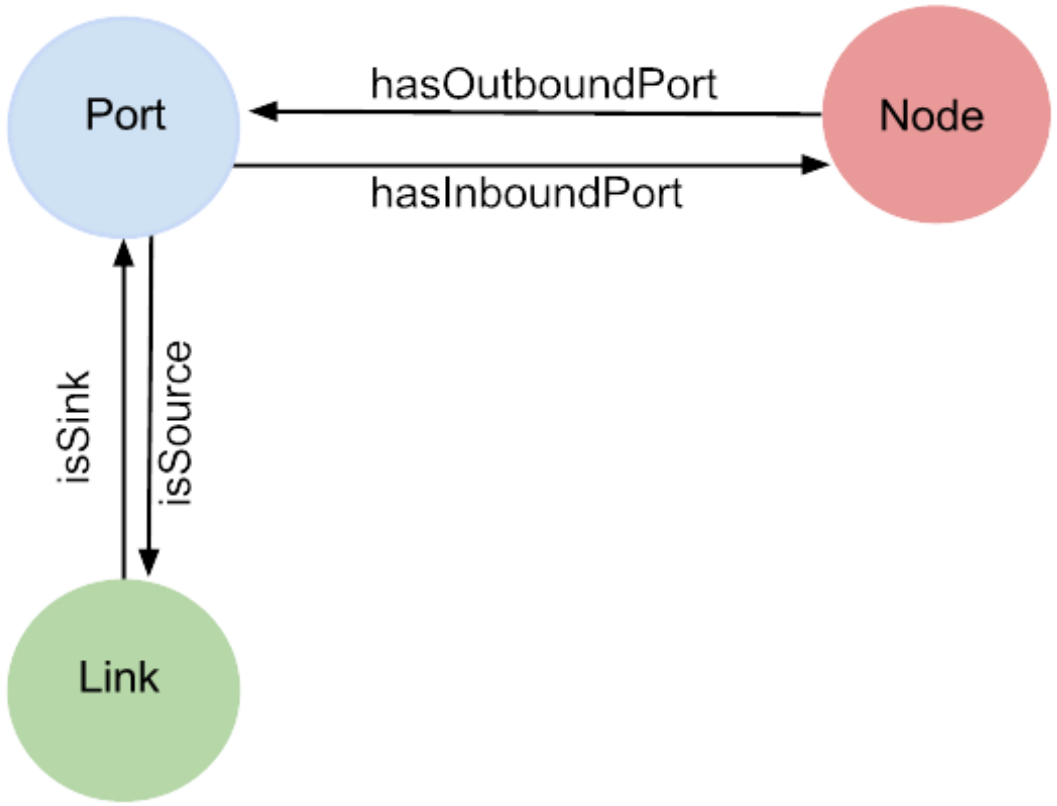
Caso de Uso
Modelagem Semântica

NML + Neo4j para primitivas Controlador SDN

Arquitetura



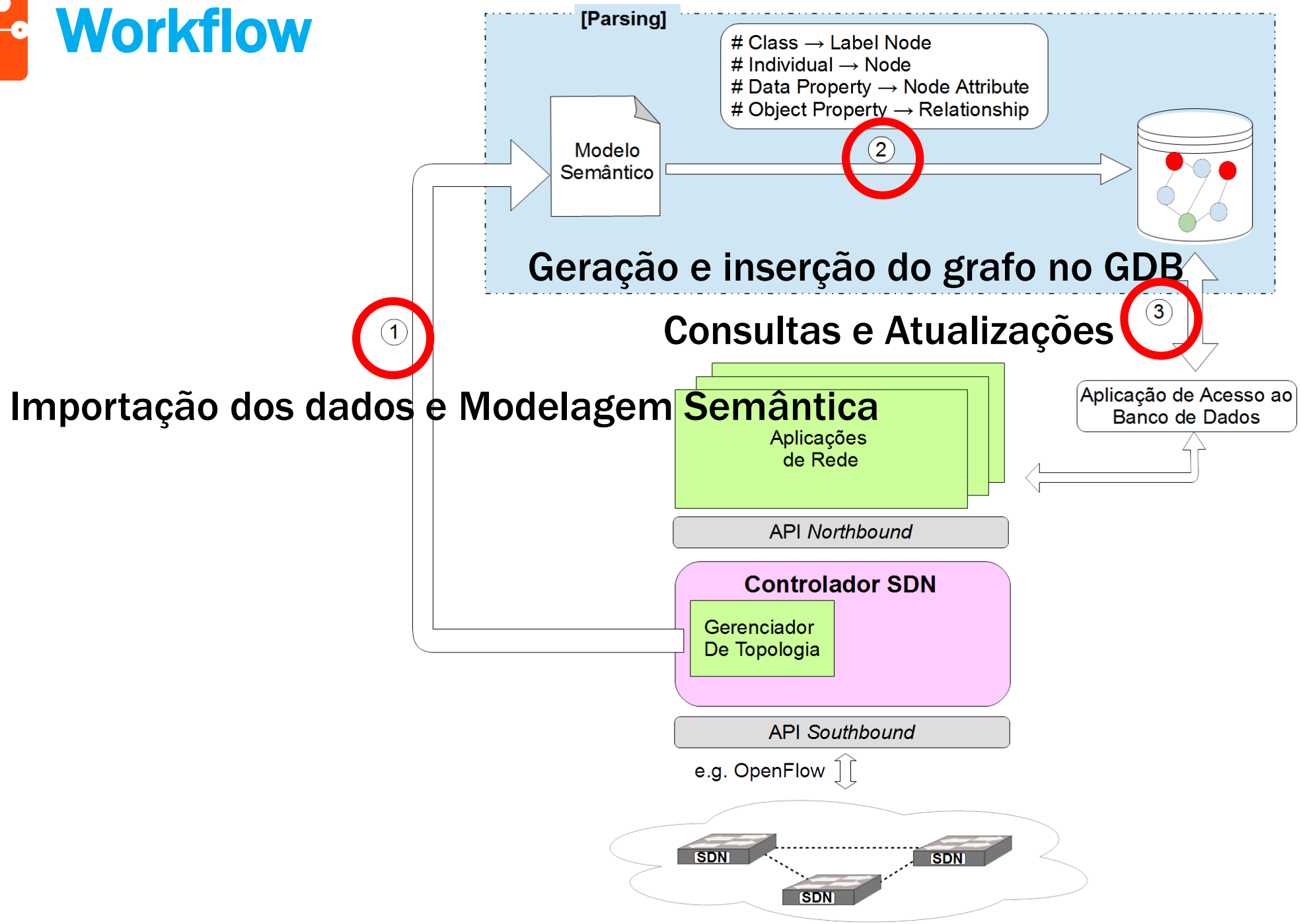
Modelo Lógico dos Dados



Análise das Primitivas
Compatibilidade das Primitivas do NetGraph

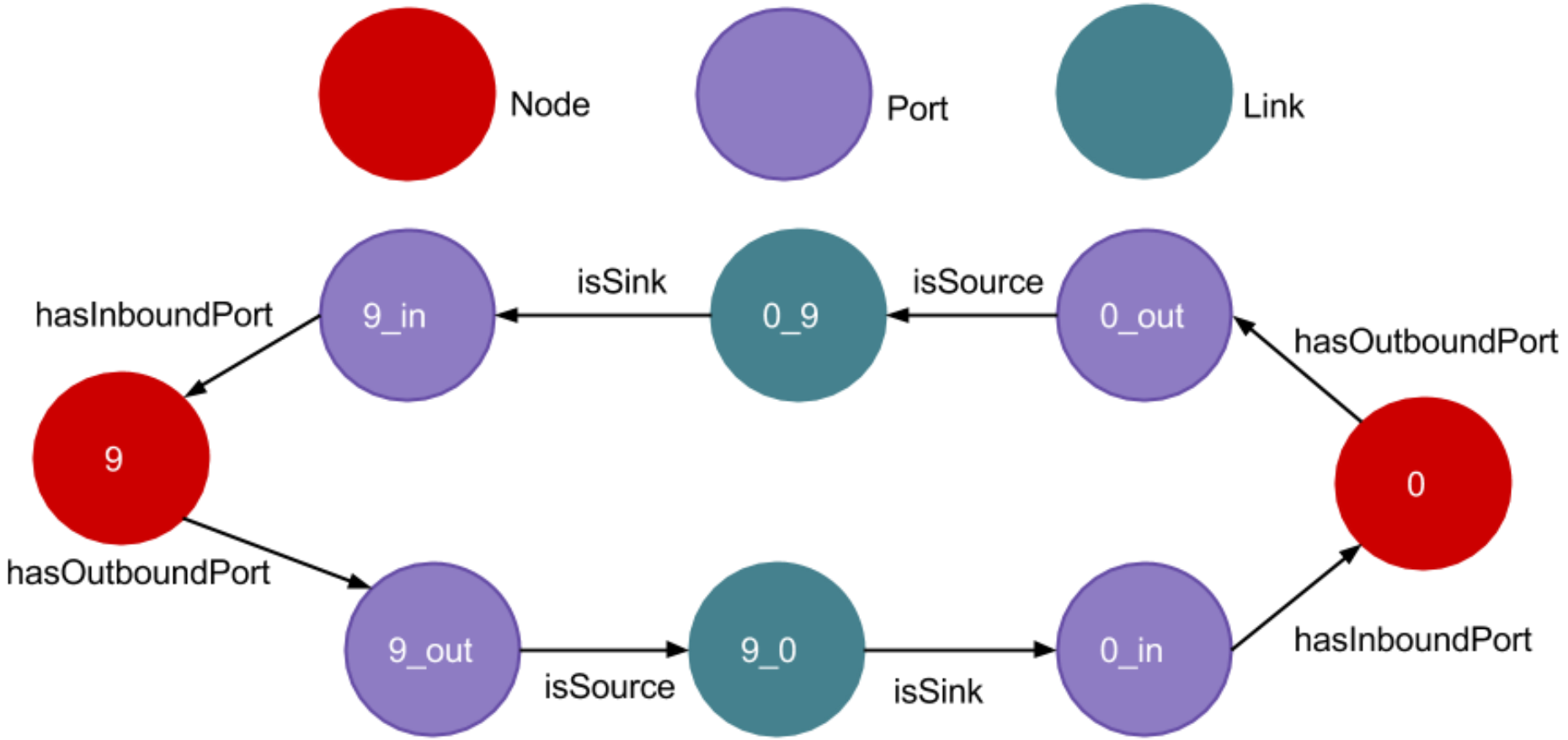
Primitiva	Modelo Semântico	GBD	Leitura/Escrita
setEdgeWeight	Não	Sim	E
getEdgeWeight	Não	Sim	L
countInDegree	Sim	Sim	L
countOutDegree	Sim	Sim	L
countNeighbors	Sim	Sim	L
computeMST	Sim	Sim	L
computeAPSP	Sim	Sim	L
computeSSSP	Sim	Sim	L
doesRouteExist	Sim	Sim	L
computeKSSSP	Sim	Sim	L
delete	Sim	Sim	E
insert	Sim	Sim	E

Workflow


[Parsing]
Class → Label Node
Individual → Node
Data Property → Node Attribute
Object Property → Relationship
Modelo Semântico
2
Geração e inserção do grafo no GDB
1
Importação dos dados e Modelagem Semântica
Consultas e Atualizações
3
Aplicação de Acesso ao Banco de Dados
Aplicações de Rede
API Northbound
Controlador SDN
Gerenciador De Topologia
API Southbound
e.g. OpenFlow
SDN
SDN
SDN

Avaliação Experimental
Modelagem dos Dados

- ✓ Exemplo de Modelagem do Relacionamento entre os Nós “9” e “0”



Topologias

- Gerador de Topologias BRITE (Universidade de Boston);

Topologias	Nós (BRITE)	Grafo Resultante
Tiny	10	76 nós (160 relacionamentos)
Small	100	640 nós (1.760 relacionamentos)
Medium	1.000	4.978 nós (11.912 relacionamentos)
Large	10.000	109.932 nós (359.728 relacionamentos)

Consultas

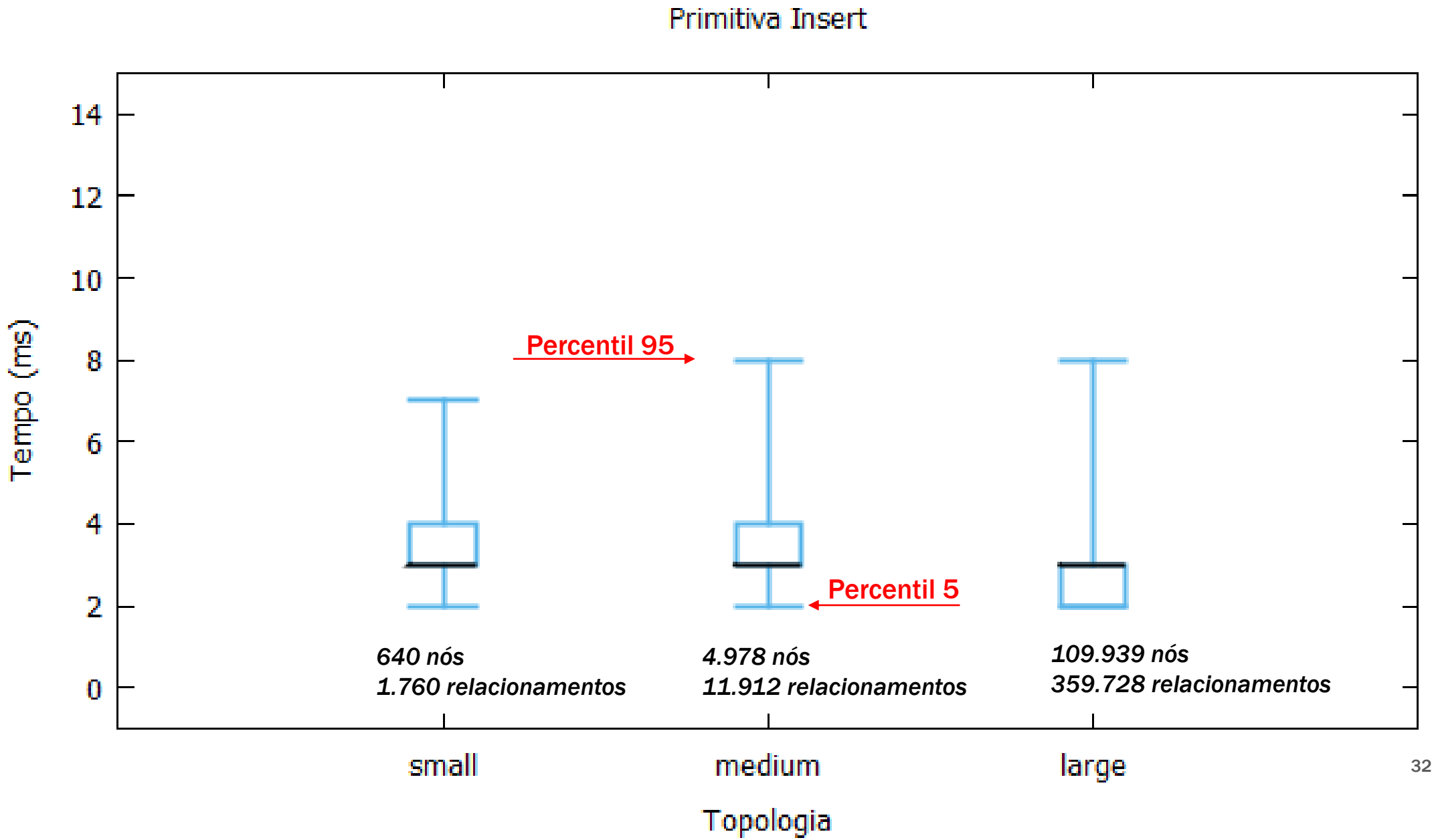
- Topologias Fixas;
- Atributos aleatórios;
- Execução de cada primitiva 1.000 vezes em cada topologia:
- Linguagem *Cypher*
  - Ex:

```
MATCH (n:Node)-[:hasOutboundPort]->(p:Port)-[isSource]->(l:Link)
WHERE n.name="A"
RETURN COUNT(l) AS CountOutDegree
```

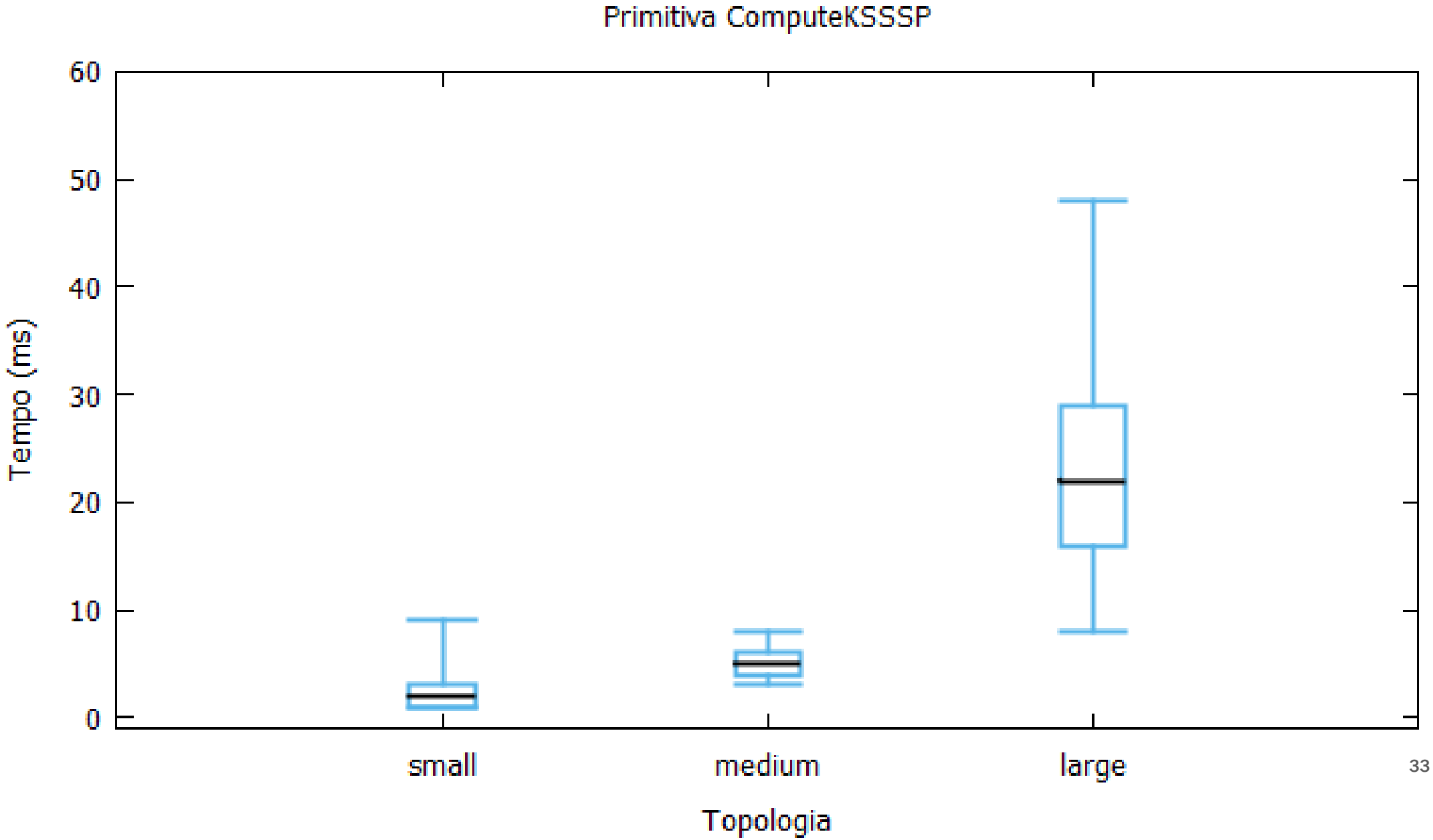
Resultados
Topologia *Large (ms)*

Primitiva	Média	Desvio Padrão	Percentil 99
setEdgeWeight	162,33	9,46	205,01
getEdgeWeight	1,70	0,74	4,00
countInDegree	854,53	146,77	1.399,05
countOutDegree	425,17	68,36	699,02
countNeighbors	4,45	2,27	10,01
doesRouteExist	37,51	29,09	73,06
computeMST	1,44	1,25	3,02
computeSSSP	5,47	4,98	29,00
computeKSSSP	26,21	37,23	81,04
computeAPSP	1,04	0,68	3,01
delete	1.053,89	162,55	1.637,02
insert	3,57	3,21	16,01

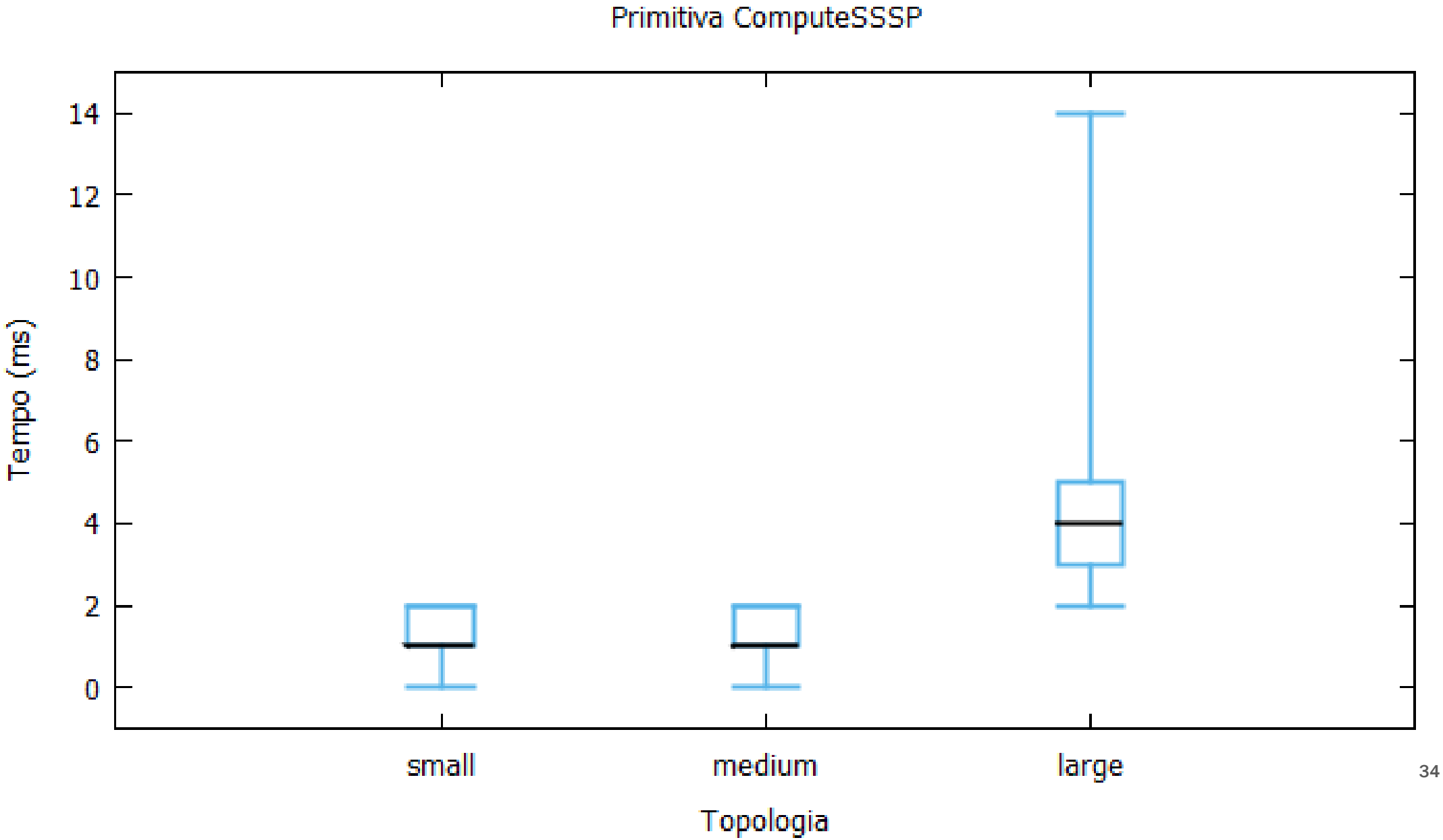
Resultados Parciais



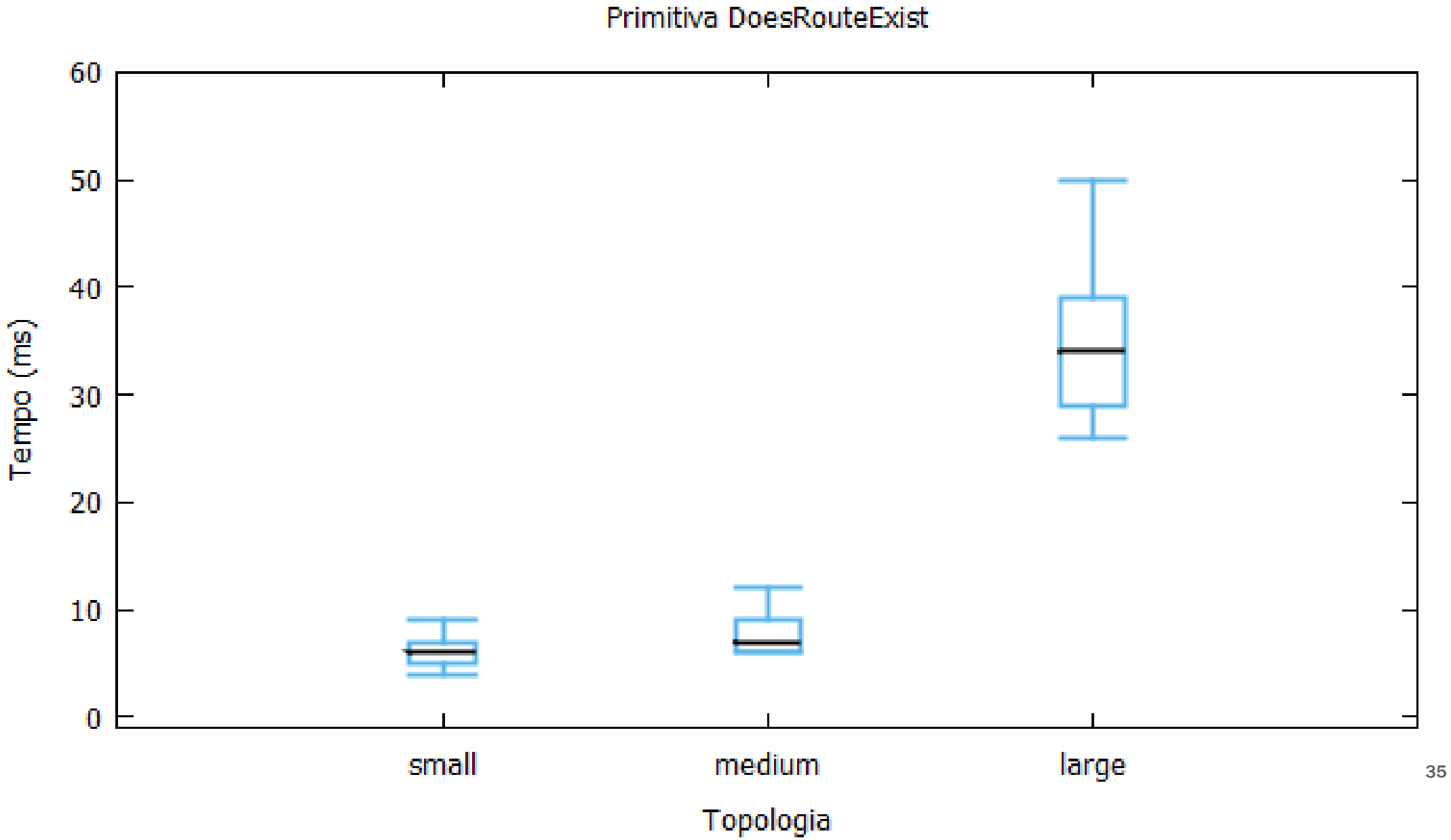
Resultados Parciais



Resultados Parciais



Resultados Parciais



Resultados Parciais
Primitivas com maior tempo de resposta

- Contagem de grau de entrada e saída (*countInDegree* e *countOutDegree*):
  - Número de *hops* (diferentes tipos de relacionamentos);

  NodeA ← ***hasInboundPort*** ← ***Port*** ← ***isSink*** ← ***Link***

- Exclusão (*delete*):
  - Maior número de *hops;*
  - Depende da conectividade do nó excluído;

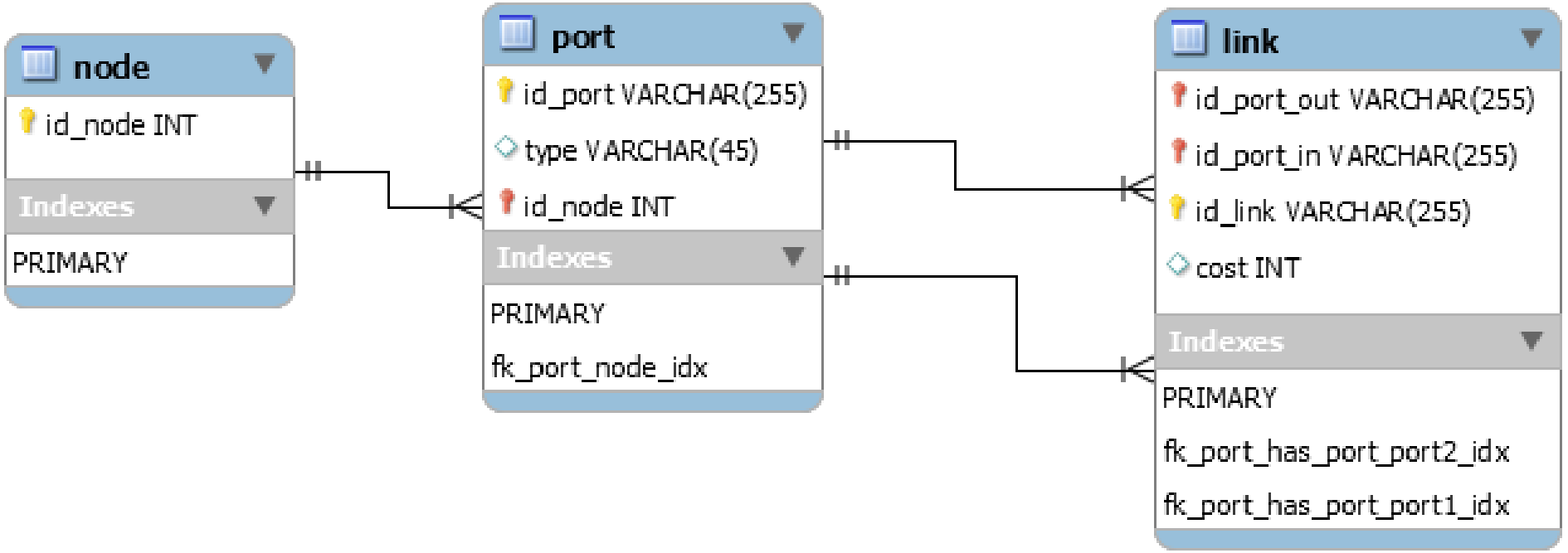
- Atribuição de peso a um *link (setEgdeWeight):*
  - Operação de leitura-escrita;

Resultados Parciais
Primitivas de Menores Caminhos

- Todos os Pares de Menores Caminhos (*computeAPSP*) menor latência que K-Menores Caminhos (*computeKSSSP*) e Menores Caminhos (*computeSSSP*);

- Otimização do GBD para cálculo de todos os pares, pois calcula entre os nós intermediários durante a travessia;

Modelagem Relacional
Modelo Entidade Relacionamento (MER)



Modelo Relacional
Resultados (MySQL) – Topologia *Large* (ms)

Primitiva	Média	Desvio Padrão	99 Percentil
countInDegree	1,39	4,57	22,02
computeSSSP	18,13	3,82	26,00
computeAPSP	2,11	1,39	7,00
delete	162,86	79,93	405,00
insert	137,36	43,80	300,00

Avaliação

Modelo Relacional

- Modelagem adaptada em tabelas;
- Para cálculo de menores caminhos foi necessário implementar/adaptar um algoritmo;
- Menor tempo de resposta:
  - CountInDegree
  - Delete

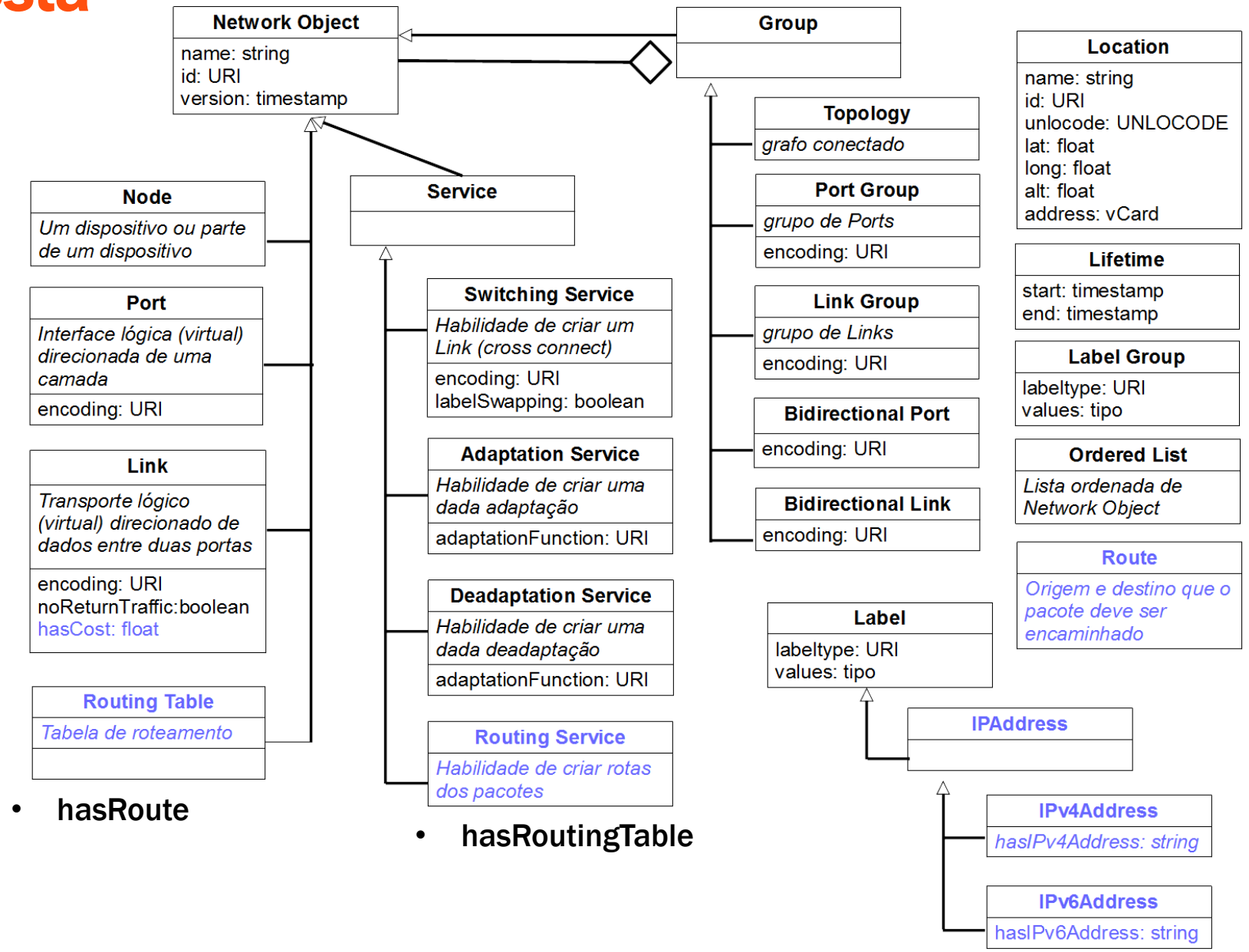
Modelo Baseado em Grafos

- Modelagem Natural;
- Suporte nativo à cálculos de menores caminhos;
- Menor tempo de resposta:
  - ComputeSSSP
  - Insert

Extensão do Modelo Semântico

- Limitações NML:
  - Representação dos dados da camada 3;
  - Protocolo IP;
  - Encaminhamento de pacotes;
  - Rotas (endereço IP origem e endereço IP destino);

Extensão do Modelo Semântico
Proposta



- hasRoute
- hasRoutingTable
- hasNextHopIPv4
- hasDstIPv4
- hasNextHopIPv6
- hasDstIPv6

Conclusões
Caso de Uso - Modelo Semântico

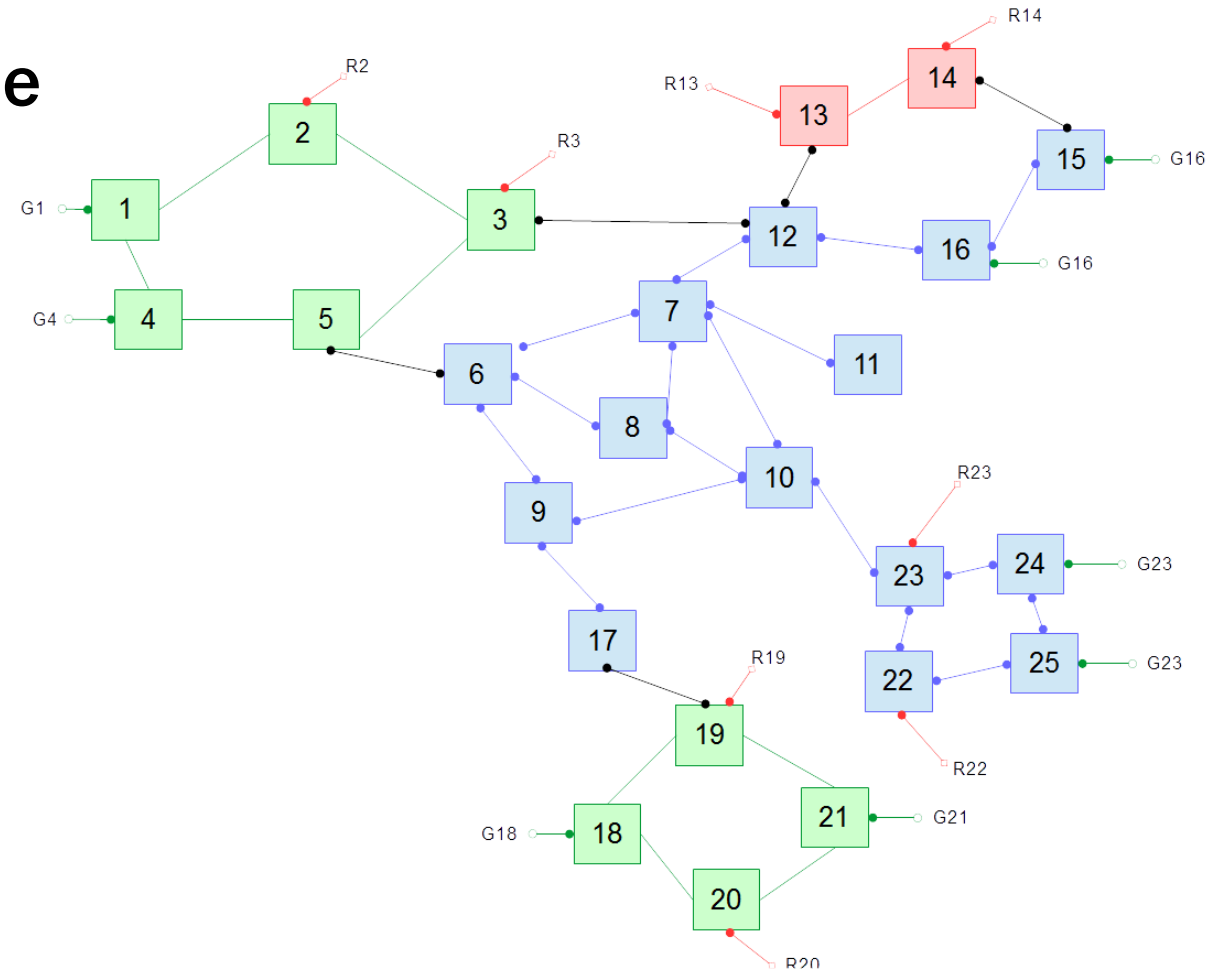
- Indexação de uma rede, conforme modelo semântico (NML) em um banco de dados baseado em grafos (Neo4j) no contexto de Redes Definidas por Software;
- Arquitetura para Integração;
- Formalização do *Parsing*:
  - Modelo Semântico →Banco de Dados
  - Banco de Dados → Modelo Semântico
- Neo4j compatível com a modelagem e linguagem Cypher flexível;
- Primitivas utilizadas por uma aplicação SDN puderam ser reproduzidas;
- Limitações do modelo semântico identificadas e estudo inicial de uma extensão do modelo NML;

Caso de Uso Multidomínios SDN

Technical Recommendation ONF-TR-502

Multidomínios SDN

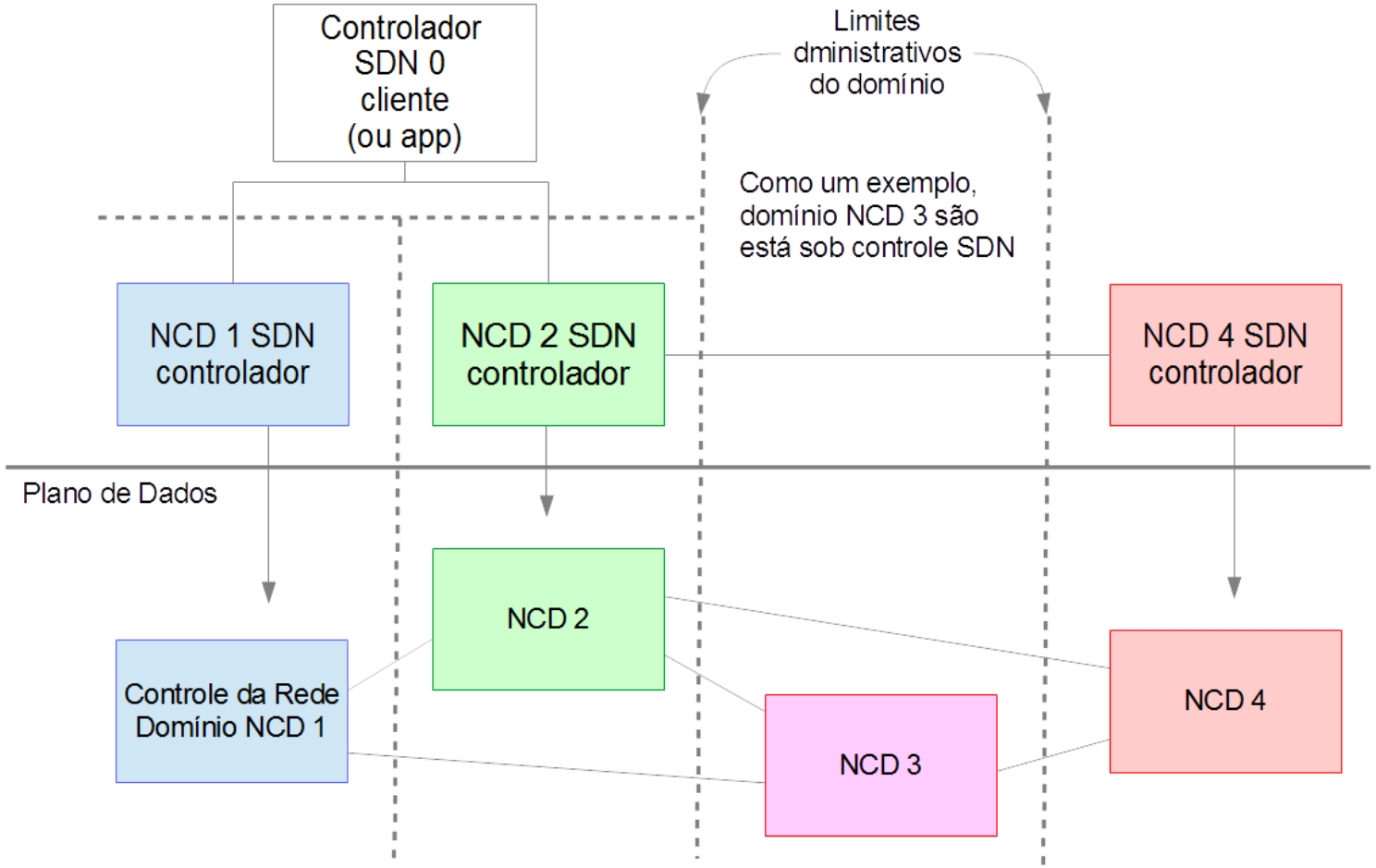
- **Múltiplos administradores possuem sua própria subrede e estão interligados**



(ONF-TR-502, 2014)

Multidomínios SDN

- Coordenação Controller to Controller (C2C)



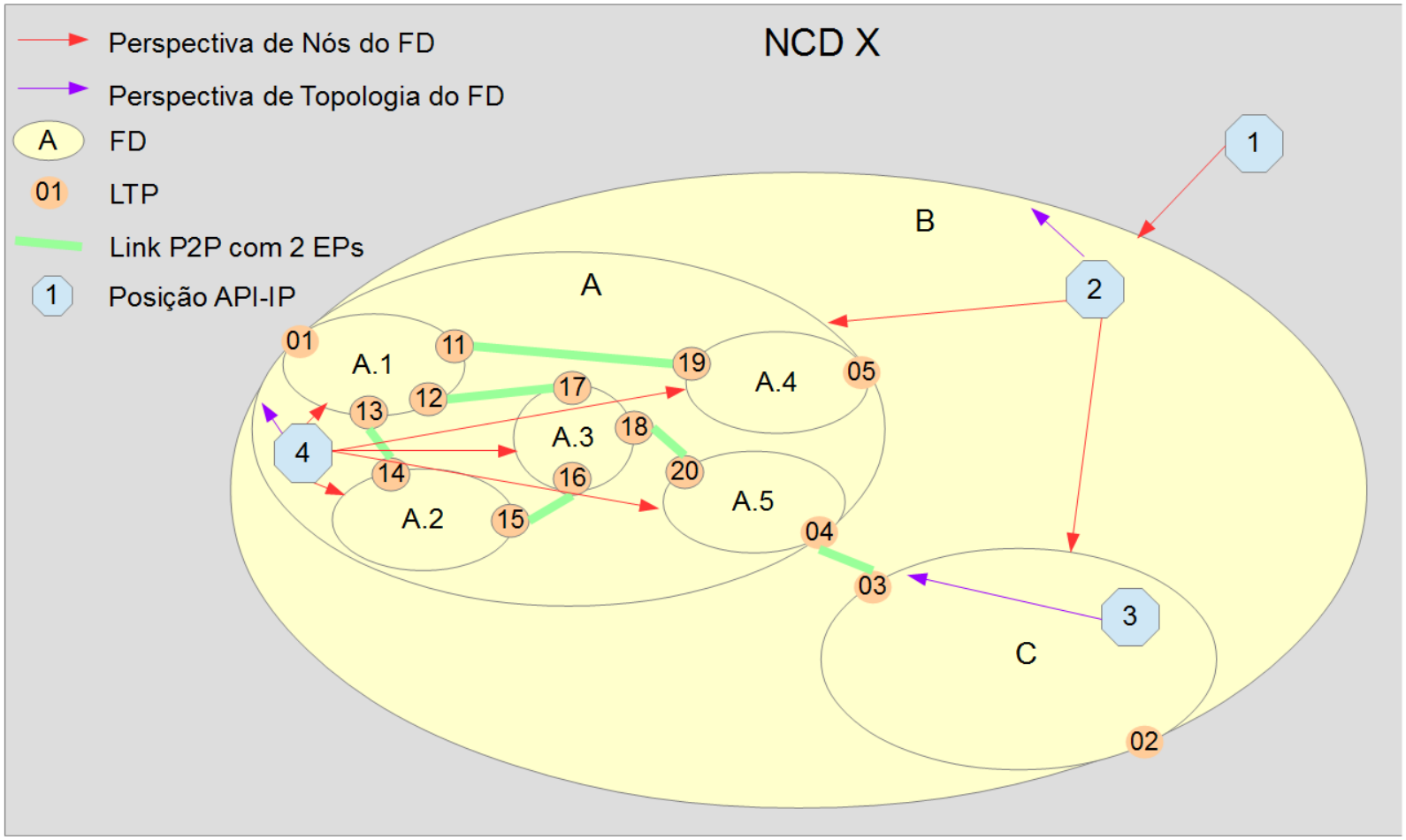
Exemplos de informações trocadas:

- **Adjacência do controlador;**
- **Informação de estados e atributos;**
- **Descoberta de topologia e vizinhos;**
- **Informação de caminhos**

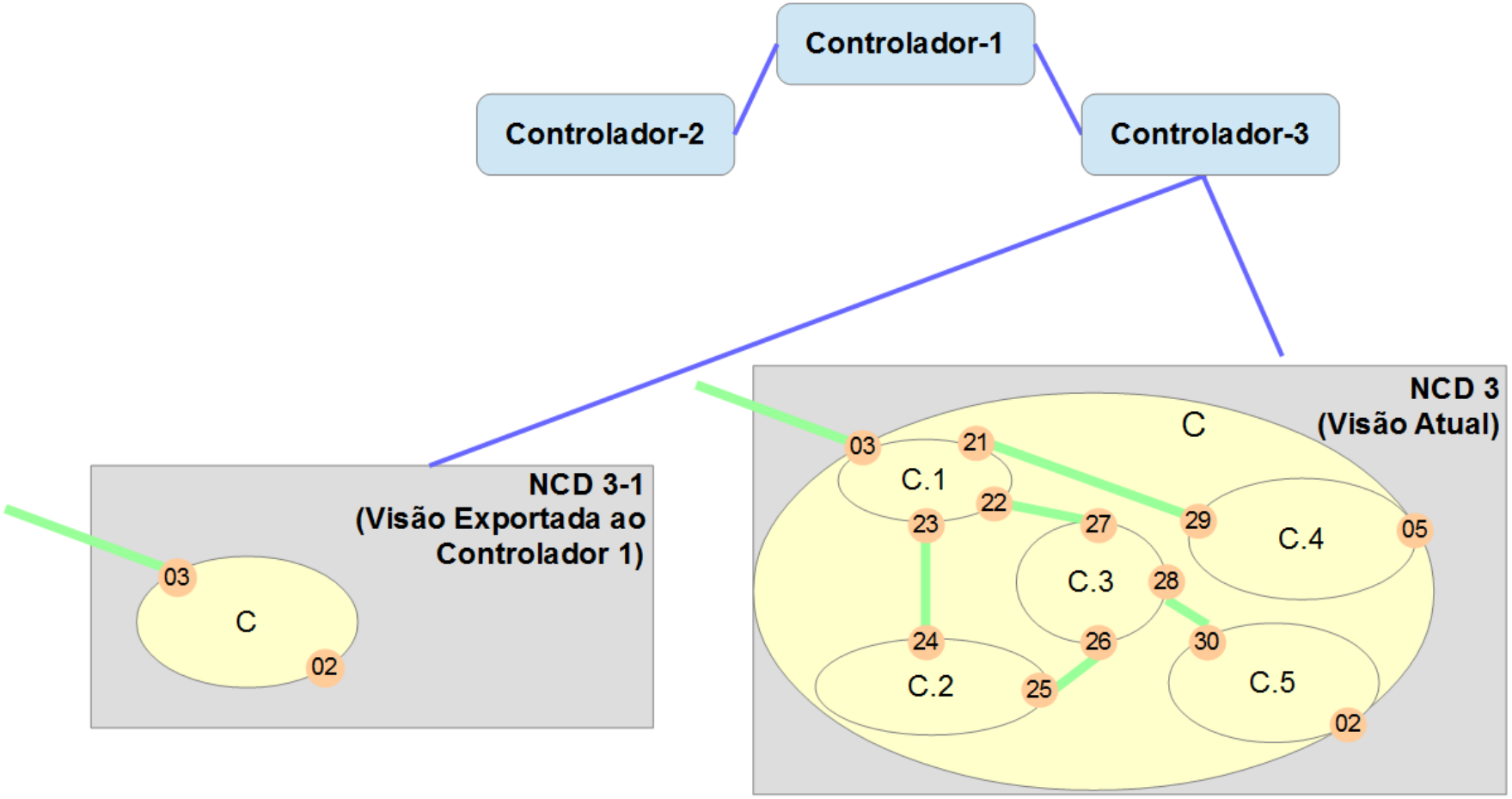
Multidomínios SDN
Modelo da Informação

- ONF-CIM (ONF-TR-512)
  - *Network Control Domain/View* (NCD): representa o escopo de controle de um controlador SDN de uma rede específica;
  - *Forwarding Domain* (FD): é a partição lógica para representar potencial para encaminhamento.
  - *Logical Termination Point* (LTP): representam portas internas e na borda de um FD.
  - *End Point* (EP): representa o acesso para encaminhamento e/ou adjacência. Um EP deve ser associado a um LTP.

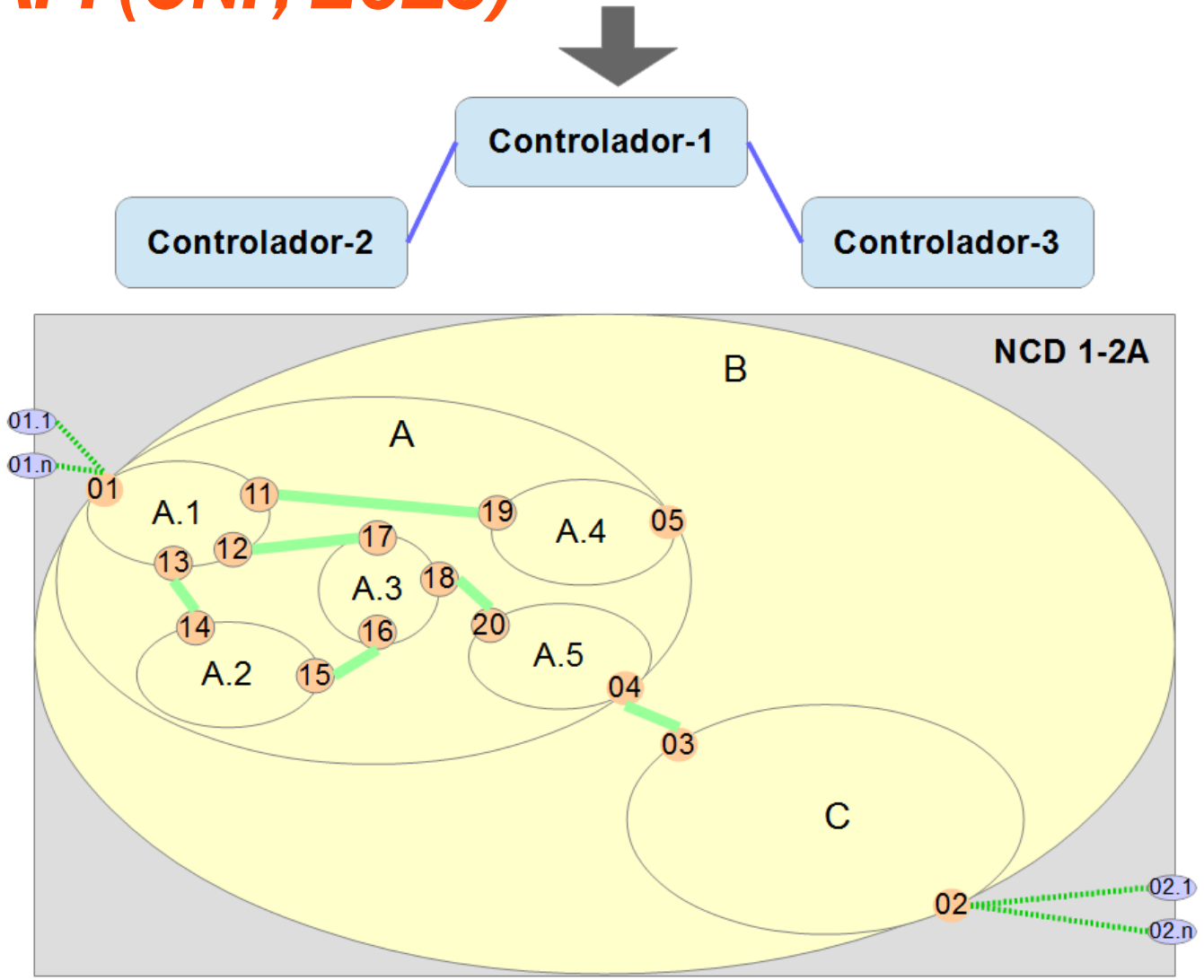
Multidomínios SDN
Transport API (ONF, 2015)



Multidomínios SDN
Transport API (ONF, 2015)



Multidomínios SDN
Transport API (ONF, 2015)

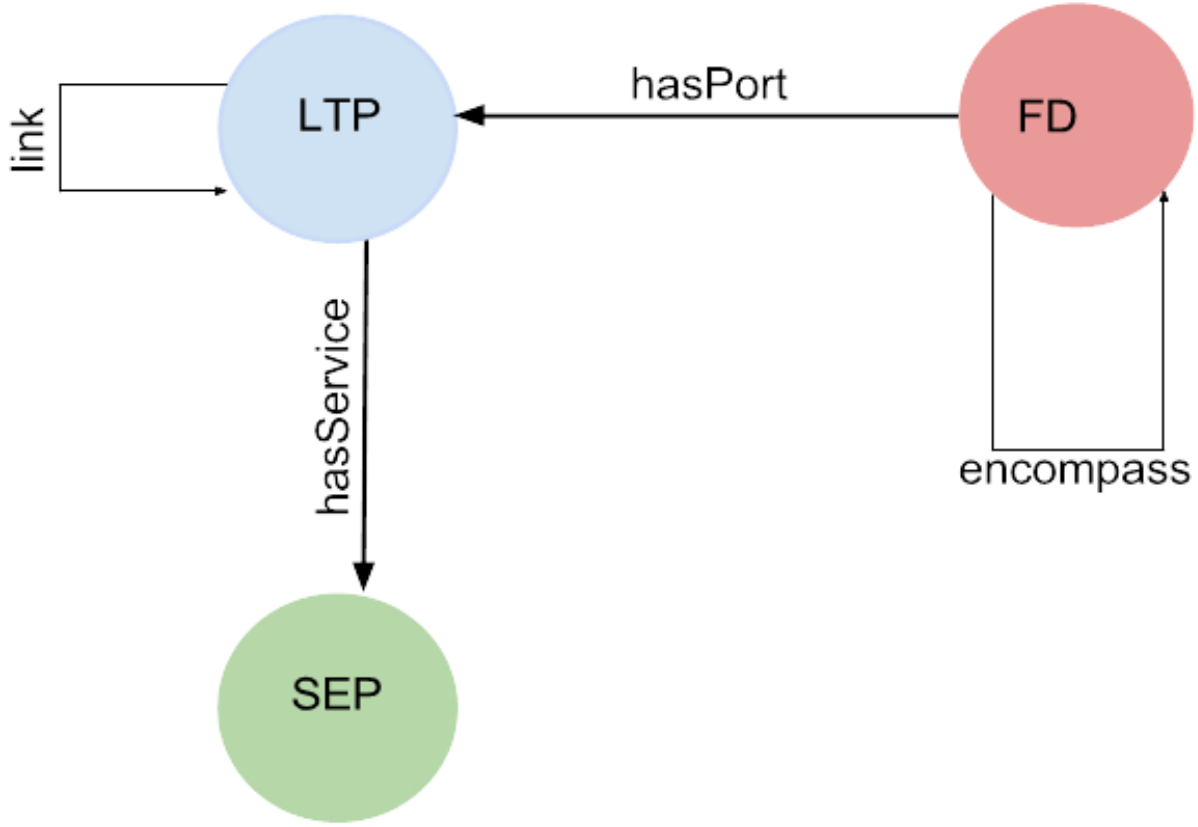


Multidomínios SDN
Subgrafos no Neo4j

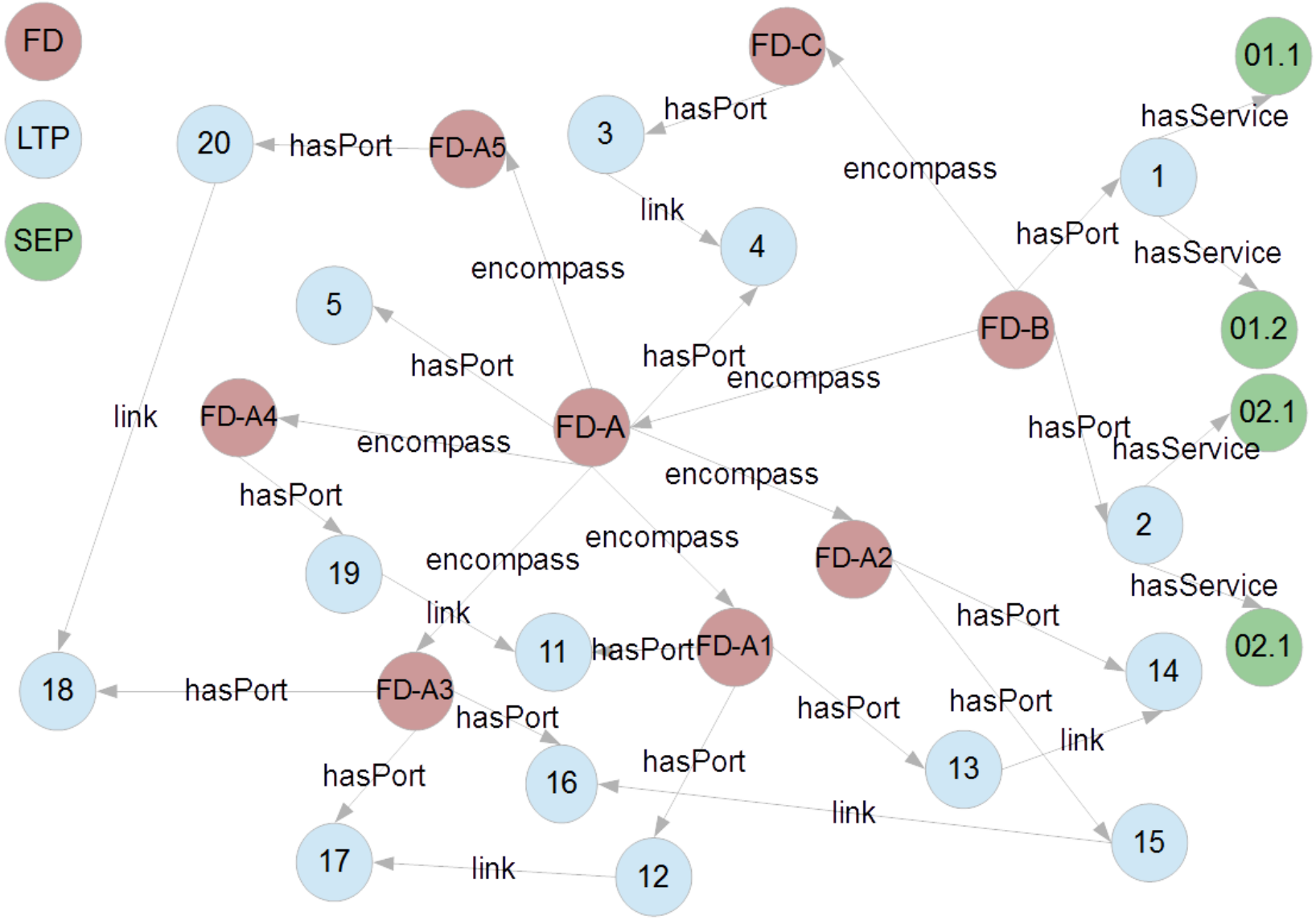
Possibilidades:

1. Consultas com restrições para buscar os subgrafos e executá-las quando necessário;
2. Nó para representar o subgrafo e relacioná-lo com os nós e relacionamentos que façam parte.

Multidomínios SDN
Modelo Lógico dos Dados



Multidomínios SDN
Exemplo



Multidomínios SDN
Primitivas Reproduzidas

- GetTopologyEncompassedByFD(FD_ID) realiza uma busca dos FDs que estão contidos em um FD;
- GetServiceEndPointDetails(FD_ID) busca quais são os SEP de um FD;
- GetNodeDetails(FD_ID) realiza uma busca de LTPs conectados a um FD;
- GetLinkDetails(FD_IDorigem, FD_IDdestino) retorna as portas de um *link* entre dois FDs.

Multidomínios SDN
Conclusões

- Foi possível reproduzir o cenário de subdomínios no Neo4j respeitando o modelo ONF-CIM;
  - O *Property Graph* atendeu as necessidades;
  - Primitivas das aplicações foram escritas na linguagem *Cypher;*
  - Consultas e Atualizações.

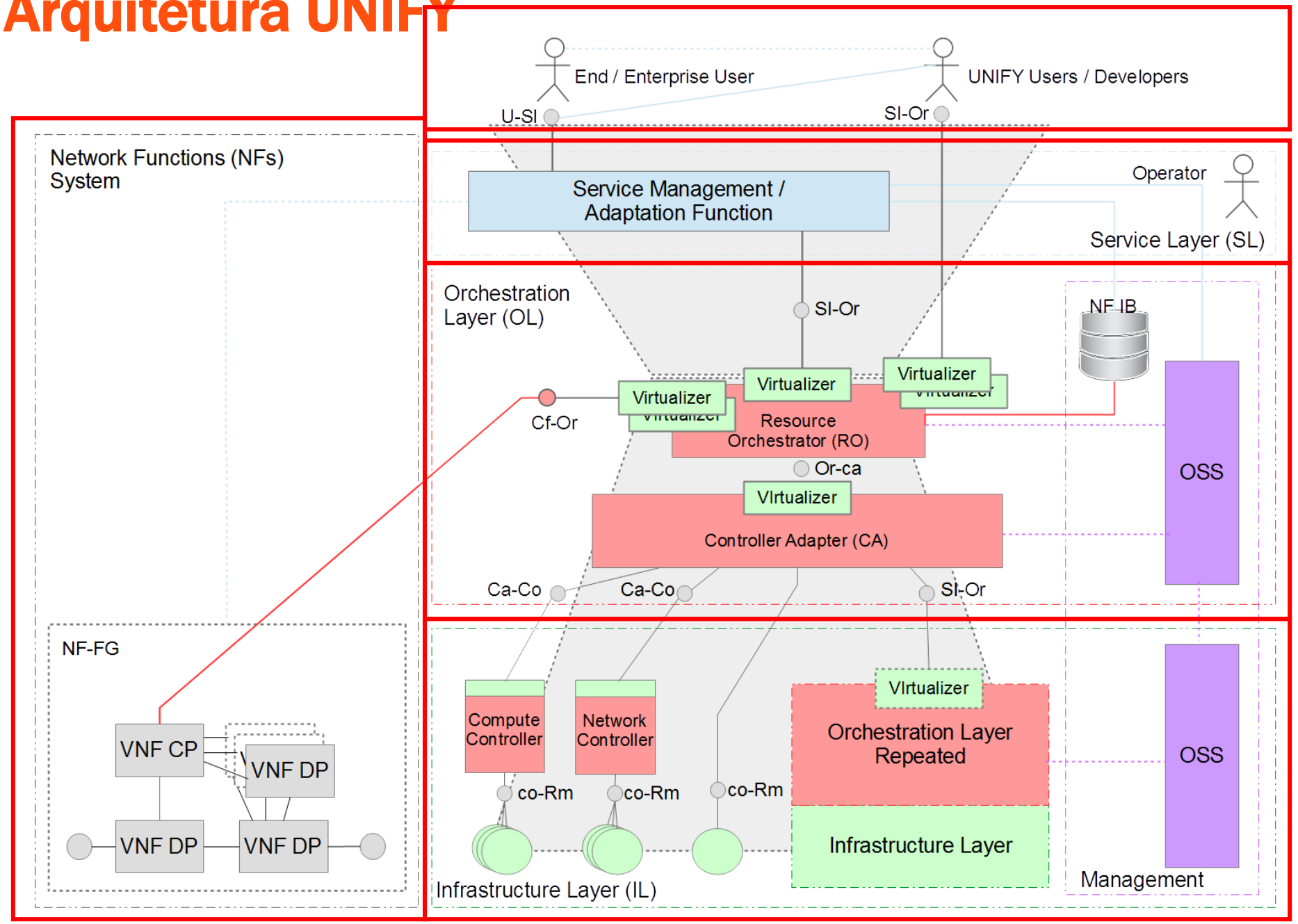
Caso de Uso
Virtualização Recursiva

Neo4j + UNIFY

Virtualização Recursiva
UNIFY

- EU FP7
  - http://www.fp7-unify.eu/
  - Provedores de serviços, fornecedores, universidades e institutos de pesquisa.
- Agregação e núcleo de redes para *data centers* para entrega de serviço;
- A virtualização de recursos pode ser multinível (tecnologias, fornecedores e domínios administrativos)

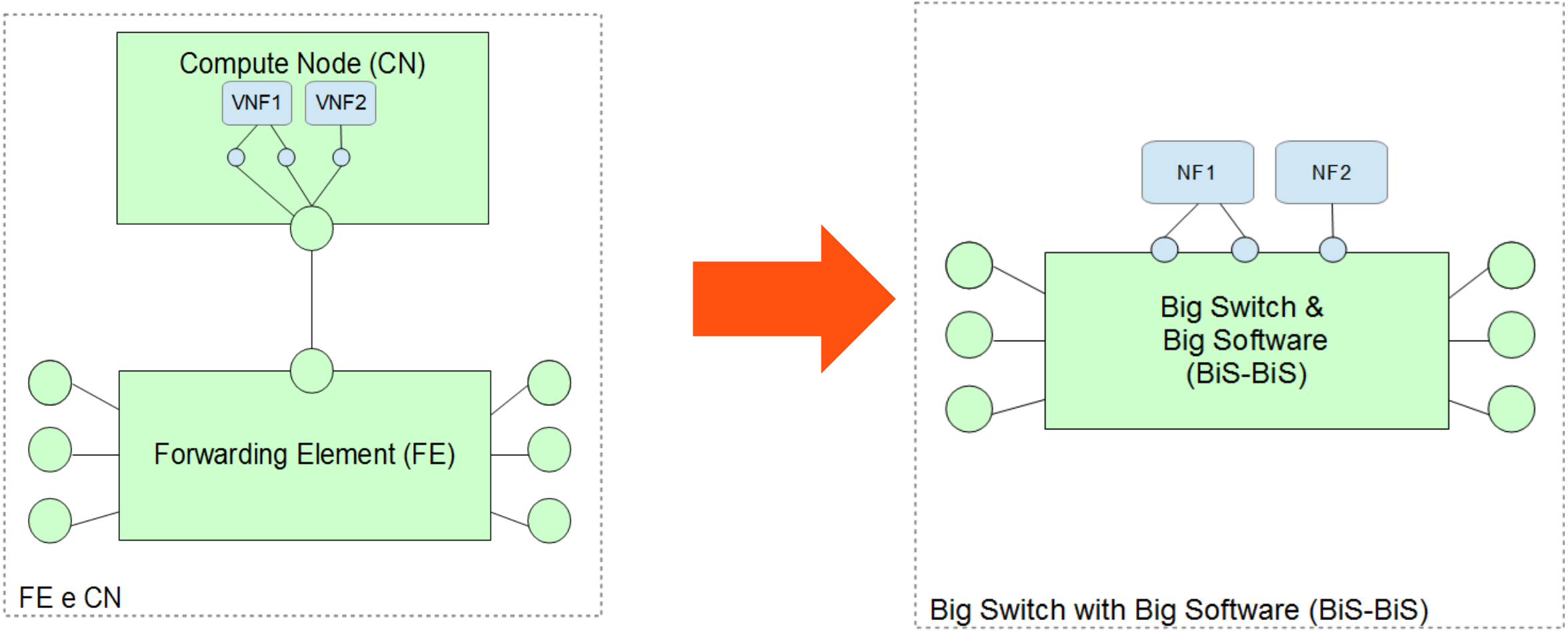
Virtualização Recursiva
Arquitetura UNIFY



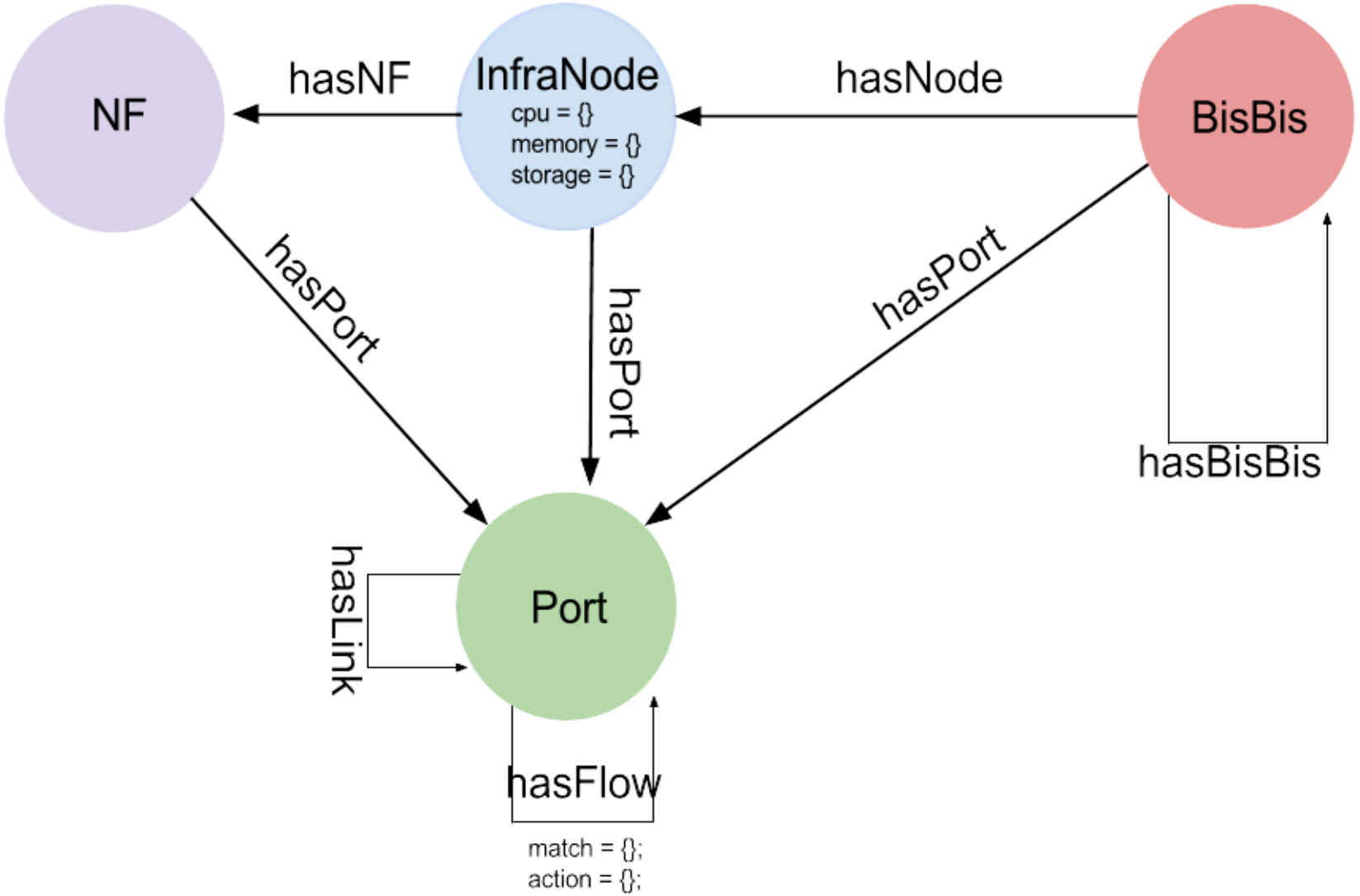
Virtualização Recursiva
Virtualizer

- Alocação das abstrações dos recursos e capacidades a um consumidor;
  - Computação
  - Armazenamento
  - Rede
  - Ambientes de execução
- A junção de *software* e abstração:
  - *Big Software with Big Switch* (BiS-BiS)

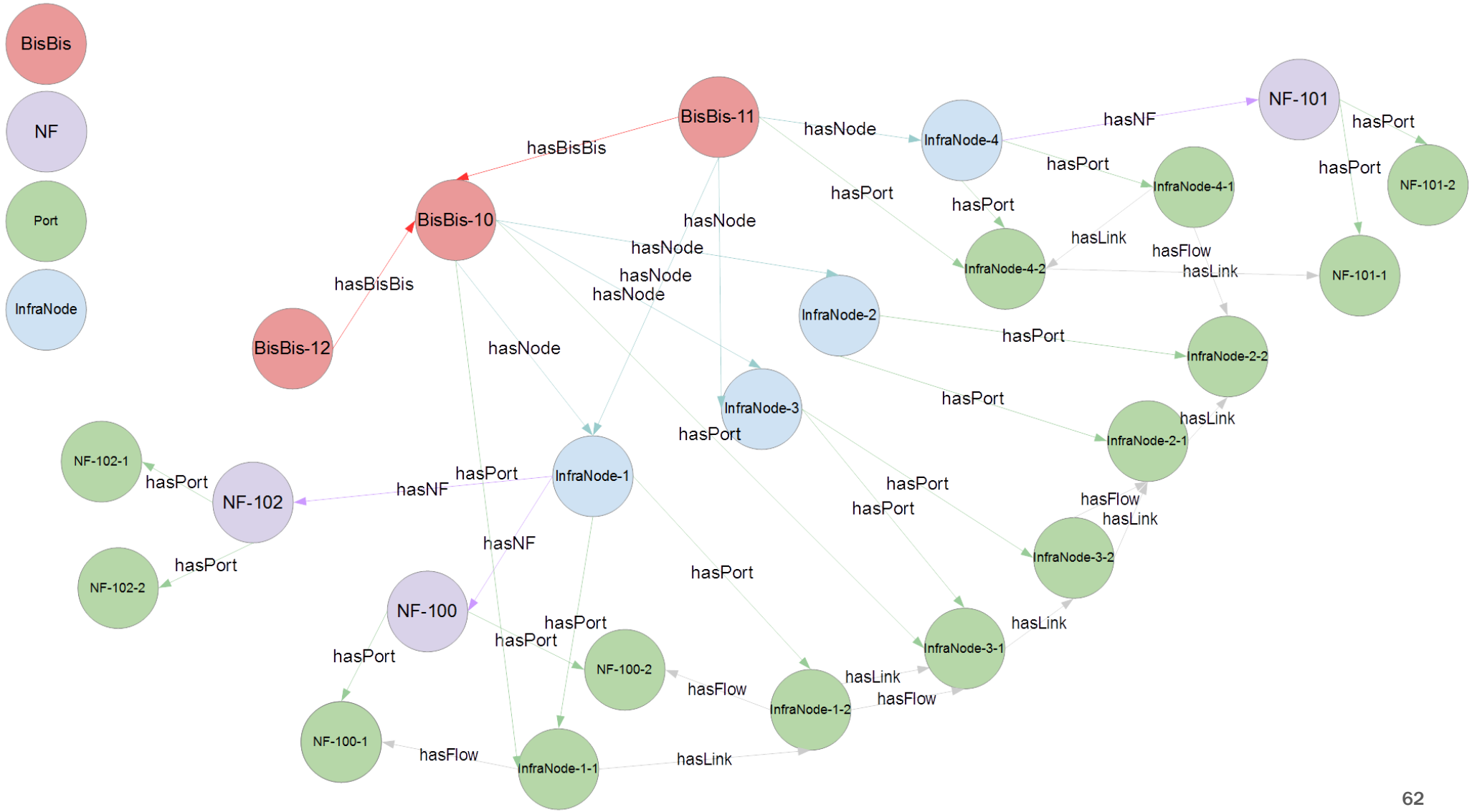
Virtualização Recursiva
BiS-BiS



Virtualização Recursiva
Modelo Lógico dos Dados



Virtualização Recursiva
Exemplo



Virtualização Recursiva
Primitivas

- Escritas na linguagem Cypher:
  - Conjunto de BiS-BiS;
  - Conjunto de InfraNodes e suas NFs;
  - Links entre portas e seus respectivos InfraNodes;
  - Alocações por *tag*;
- Geração de BiS-BiS
  - (*i*) todos os recursos de infraestrutura
  - (*ii*) as portas com apenas um *link* (de entrada ou de saída)

Virtualização Recursiva
Conclusões

- Foi possível reproduzir o cenário no Neo4j;
  - O *Property Graph* atendeu as necessidades;
  - Primitivas do consumo do orquestrador foram escritas na linguagem *Cypher;*
  - Consultas e Atualizações;
- Primitiva de alocação por *tag:*
  - Relacionamento *hasFlow: match e action*

Agenda

- Introdução
- Objetivos
- Fundamentação Teórica
  - Redes Definidas por Software
  - Virtualização de Funções de Rede
  - Modelos Semânticos
  - Bancos de Dados Baseados em Grafos
- Casos de Uso
  - Modelagem Semântica
  - Multidomínios SDN
  - Virtualização Recursiva
- **Conclusão e Trabalhos Futuros**

Conclusões

- Modelagem Semântica
  - Arquitetura, *Workflow*, *Parsing*, Primitivas SDN, Limitações identificadas e extensão proposta;
- Multidomínios SDN
  - Cenário de troca de visões entre controladores, Primitivas;
- Virtualização Recursiva
  - Cenário do UNIFY, Primitivas;
- Bancos de Dados Baseados em Grafos atendeu os três cenários

Trabalhos Futuros

- Avaliar o desempenho com cargas de trabalho dinâmicas e aplicações no controlador OpenDaylight usando REST APIs;
- Explorar otimizações na latência e capacidade do sistema via pré-calculo e uso de propriedades para habilitar ou desabilitar nós no grafo;
- Validar a proposta de extensão do NML com as tabelas de roteamento;
- Explorar novos casos de uso do UNIFY (GDB e Modelo Semântico)
- Desenvolvimento de extensões do modelo semântico (NML/INDL) para redes SDN e funções de rede virtualizadas (NFV);
- Explorar as extensões do modelo semântico com *reasoners*, realizar inferências e verificar inconsistências;

Publicações

- ”*Towards Semantic Network Models via Graph Databases for SDN Application*”
  - Com Mateus A. S. Santos, Luciano B. de Paula e Christian E. Rothenberg
  - *4th European Workshop on Software Defined Networks (EWSDN)* – Bilbao-Espanha - Outubro de 2015.
- ”Modelos Semânticos em Bancos de Dados Baseados em Grafos para Aplicações de Controle de Redes Definidas por Software”
  - Com Mateus A. S. Santos, Luciano B. de Paula e Christian E. Rothenberg
  - XX Workshop de Gerência de Redes e Serviços (WGRS) do XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) - Vitória-ES - Maio de 2015.

Thanks! Obrigado! (More) Questions?

https://github.com/intrig-unicamp/NML-Neo4j

cypriano@dca.fee.unicamp.br

Backup

Modelos Semânticos

Exemplo RDF (Sintaxe XML/RDF)

```
<?xml version="1.0"?>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#" >

 <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
           <contact:fullName>João Silva</contact:fullName>
           <contact:mailbox rdf:resource="mailto:joao@exemplo.org"/>
           <contact:personalTitle>Dr.</contact:personalTitle>
 </contact:Person>
</rdf:RDF>
```

Controladores SDN

- Controlador ONIX (Koponen et al., 2010):
  - Implementa plano de controle para redes SDN;
  - Distribui problemas de manutenção das informações entre diferentes controladores;
- Controlador ONOS (Berde et al., 2014):
  - Princípios do ONIX;
  - Dois protótipos (GDBs e processamento em memória).
- Grafos em SDN (Pantuza et al., 2014):
  - Suporte à representação dinâmica da rede;
  - Árvore de custo mínimo em tempo real sobre o grafo da rede;
- Biblioteca NetGraph (Raghavendra et al., 2012):
  - Atualizações periódicas do estado da rede;
  - Resultados de Consultas para controlador SDN;
- *Showed Subgraph* (Lauer et al., 2013):
  - Troca de informações entre controladores;
  - Informações da topologia da rede (estado, políticas, métricas);

Controladores SDN

- Comparação das propostas de Modelos e Bancos de Dados das propostas:

Trabalho	Modelo de Dados	Banco de Dados
Onix (Koponen et al., 2010)	NIB	Base de dados transacional
POX Adaptado (Pantuza et al., 2014)	NOM	Não aplica
NetGraph (Raghavendra et al., 2012)	XML	Não aplica
ONOS (Berde et al., 2014)	Modelo próprio	GDB e Estrutura de Dados Otimizada
Showed Subgraphs (Lauer et al., 2013)	Modelo próprio	GDB DEX

Algoritmos *Parsing*

Pseudocódigo 4.1 Geração do Modelo Semântico

enquanto !Fim do documento **faça**
  leia atual;
  se nó **então**
    crie um *Individual* da classe *Node*, *URI* ← identificador ;
    crie um *DatatypeProperty* do tipo *name*;
    crie um *Individual* da classe *Port*, *URI* ← identificador + "*_in*";
    crie um *Individual* da classe *Port*, *URI* ← identificador + "*_out*";
    crie um *ObjectProperty* do tipo *hasInboundPort* entre o *Node* e a porta de entrada;
    crie um *ObjectProperty* do tipo *hasOutboundPort* entre o *Node* e a porta de saída;
  senão
    crie um *Individual* da classe *Link*, *URI* ← identificadorNó1+"_"+identificadorNó2;
    crie um *ObjectProperty* do tipo *isSource* entre o *Link* e a nó1*_out*;
    crie um *ObjectProperty* do tipo *isSink* entre o *Link* e a a nó2*_in*;
    crie um *Individual* da classe *Link*, *URI* ← identificadorNó2+"_"+identificadorNó1;
    crie um *ObjectProperty* do tipo *isSource* entre o *Link* e a nó2*_out*;
    crie um *ObjectProperty* do tipo *isSink* entre o *Link* e a a nó1*_in*;
  fim se
fim enquanto

Algoritmos *Parsing*

Pseudocódigo 4.2 Inserção do Grafo no GDB

```
enquanto !Fim do documento faça
    leia atual;
    se Individual então
        crie um nó, label leftarrow ClassName, id ← URI;
        para cada Datatype Properties faça
            atributo ← Datatype Property, valor ← Value;
        fim para
        para cada Object Properties faça
            crie um relacionamento, tipo ← Object Property, com Individual;
        fim para
    fim se
fim enquanto
```

Algoritmos *Parsing*

Pseudocódigo 4.3 Geração do modelo a partir do GDB

```
enquanto !Fim do documento faça
    leia atual;
    se nó então
        crie um Individual, Class ← label, URI ← id ;
        para cada Atributos faça
            crie uma data property, Value ← valor;
        fim para
        para cada Relacionamentos faça
            crie uma object property, Individual ← nóRelacionado;
        fim para
    fim se
fim enquanto
```

Consultas – Primitivas NetGraph

- **Grau de entrada de um *Node*:**

```
1. MATCH (n:Node)<-[:hasInboundPort]-(p:Port)<-[isSink]-(l:Link)
2. WHERE n.name={nameNode}
3. RETURN COUNT(l) AS CountOutDegree
```

- **Grau de saída de um *Node*:**

```
1. MATCH (n:Node)-[:hasOutboundPort]->(p:Port)-[isSource]->(l:Link)
2. WHERE n.name={nameNode}
3. RETURN COUNT(l) AS CountOutDegree
```

- **Vizinhos de um *Node*:**

```
1. MATCH (n:Node)-[]-(p:Port)-[]-(l:Link)-[]-(p1:Port)-[]-(n2:Node)
2. WHERE n.name={nameNode}
3. RETURN DISTINCT(n2) AS Neighbors
```

Consultas – Primitivas NetGraph

- **Verificação da existência de rota entre dois *Nodes*:**

```
1. MATCH p=shortestPath((n:Node)-[*]-(m:Node))
2. WHERE n.name = {nameNode} AND m.name={nameNode1}
3. RETURN COUNT(p) >0 AS DoesRouteExist
```

- **Cálculo de menor caminho de um *Node* para todos os outros:**

```
1. MATCH (n:Node),(m:Node), p=shortestPath((n)-[*]->(m))
2. WHERE n.name={nameNode}
3. RETURN p
```

- **Cálculo de menor caminho de todos os pares de *Node*:**

```
1. MATCH p=shortestPath((n:Node)-[*]->(m:Node))
2. RETURN p
```

Consultas – Primitivas NetGraph

- **Cálculo de *k* menores caminhos entre dois *Nodes*:**

```
1. MATCH (n:Node),(m:Node), p=allShortestPaths((n)-[*]-(m))
2. WHERE n.name={nameNode} AND m.name={nameNode1}
3. RETURN p LIMIT {valueK}
```

- **Cálculo de *Minimum Spanning Tree* a partir de uma *Node* de origem:**

```
1. MATCH p=shortestPath((n:Node)-[*]->(m:Node))
2. WHERE n.name = {nameNode}
3. RETURN p AS MinimumSpanningTree
```

- **Exclusão de um *Node* (e suas *Ports*):**

```
1. MATCH (n:Node)-[r1]-(p:Port)-[r2]-(l:Link)-[r3]-(p2:Port)
2. WHERE n.name={nameNode}
3. DELETE n,r1,p,r2,l,r3
```

Consultas – Primitivas NetGraph

- **Atribuição de custo a um *Link*:**

```
1. MATCH (n:Node)-[:hasOutboundPort]-()-[r]-(l:Link)
2. WHERE n.name={nameNode} AND l.name={nameLink}
3. SET l.cost={valueCost}
```

- **Busca de custo de um *Link*:**

```
1. MATCH (n:Node)-[:hasOutboundPort]-()-[r]-(l:Link)
2. WHERE n.name={nameNode} AND l.name={nameLink}
3. RETURN l.cost
```

- **Inclusão de um *Node* (e suas *Ports*):**

```
1. CREATE (n1:Node{name: newNode})
2. CREATE (n2:Port{name: portInNewNode})
3. CREATE (n3:Port{name: portOutNewNode})
4. WITH n1, n2, n3
5. CREATE (n1)<-[r:hasInboundPort]-(n2)
6. CREATE (n1)-[r2:hasOutboundPort]->(n3)
7. RETURN r,r2
```

Consultas – Multidomínios SDN

- **Topologias contidas em um FD:**

```
1. MATCH (n:FD)-[r:encompas]-(m)
2. WHERE n.name={nameFD}
3. RETURN m AS getTopologyEncompassedByFD
```

- **Detalhes (portas) de um FD:**

```
1. MATCH (n:FD)-[r:hasPort]-(m)
2. WHERE n.name={nameFD}
3. RETURN m AS getNodeDetails
```

- **Detalhes (*links*) entre dois FDs:**

```
1. MATCH (n:FD)-[:hasPort]-(o)-[r]-(p)-
[:hasPort]-(m:FD)
2. WHERE n.name = {nameFD1} AND m.name =
{nameFD2}
3. RETURN o AS DetailsLink1, p AS DetailsLink2
```

- **Detalhes (*service endpoints*) de um FDs:**

```
1. MATCH (n:FD)-[:hasService]-(m)
2. WHERE n.name = {nameFD}
3. RETURN m AS getServiceEndPointDetails
```

Consultas – Virtualização Recursiva (UNIFY)

- **Conjunto de BisBis e seus *InfraNodes*:**

```
1. MATCH (n:BisBis)-[:hasNode]-(m:InfraNode)
2. RETURN n AS BisBis, m AS InfraNode
```

- **Conjunto de *InfraNodes* e suas NFs:**

```
1. MATCH (n:InfraNode)-[:hasNF]-(m:NF)
2. RETURN n AS InfraNode, m AS NF
```

- ***Links* entre portas e seus respectivos *InfraNodes*:**

```
1. MATCH (i:InfraNode)-[:hasPort]->(n:Port)-[:hasLink]-(m)<-[:hasPort]-
(j:InfraNode)
2. Return i AS InfraNode1, n AS Port1, m AS Port2, j AS InfraNode2
```

Resultados Parciais
Topologia *Small (ms)*

Primitiva	Média	Desvio Padrão	Percentil 99
setEdgeWeight	8,78	3,23	23,02
getEdgeWeight	1,73	0,76	3,00
countInDegree	17,94	11,36	65,01
countOutDegree	8,35	3,46	23,00
countNeighbors	6,16	22,43	14,07
doesRouteExist	6,55	3,82	15,02
computeMST	1,12	0,66	2,00
computeSSSP	1,34	1,38	4,00
computeKSSSP	2,94	3,44	12,00
computeAPSP	1,04	0,84	4,01
delete	20,71	7,20	48,01
insert	3,66	3,26	15,02