Federation-less-federation of Network-virtualization Platforms

Yasusi Kanada, Toshiaki Tarui, and Kei Shiraishi
Hitachi, Ltd.

Introduction

- **We are developing VNode and VNode Platform in a collaborative project.**

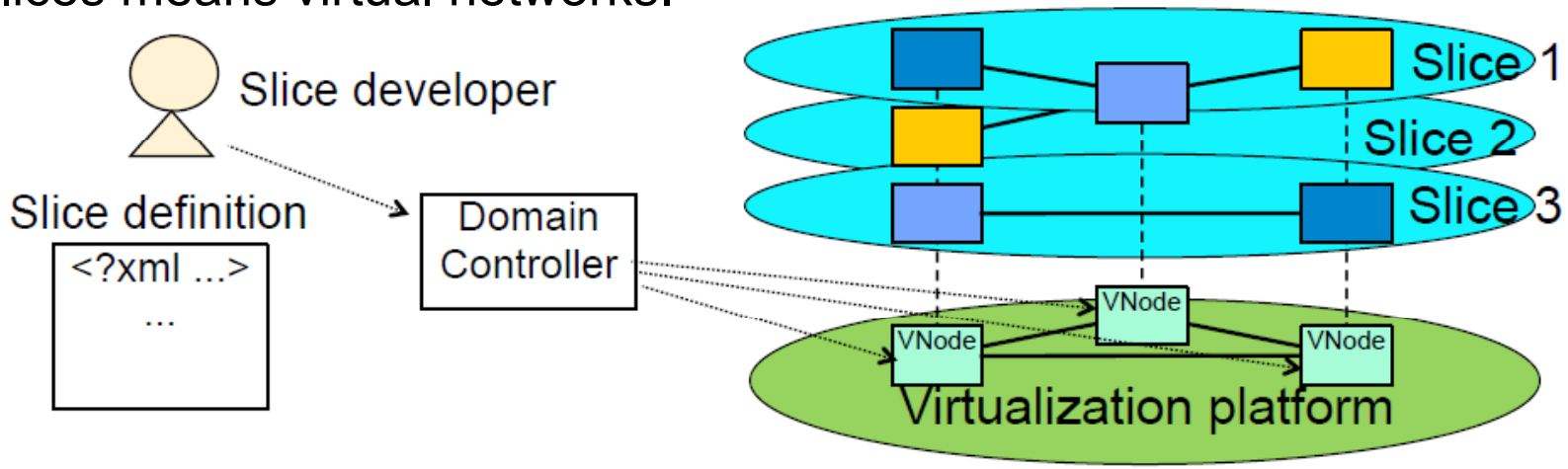
- **VNode** is a *deeply-programmable* physical node with network-virtualization function.
- **Deeply-programmable**: packet data processing, such as new *non-IP* protocol processing, can be programmable.
- **VNode Platform** is a virtualization platform, which enables concurrent creation and use of multiple slices (virtual networks).

Research Goals

- **Research goals** are **federation among two or more virtualization platforms including the VNode Platform.**
- **Final goal: Heterogenious federation**
  - To federate *new* VNode Platform and several other platforms including ProtoGENI (developed in GENI Project in US).
- **First goal: Homogenious federation of VNode Platforms**
  - To federate two or more *previously-developed* VNode Domains.
    -- This VNode Platform does *not* have federation functions.
  - To enable *Non-IP* data communication on a cross-domain slice.

Network Virtualization Platform and VNode

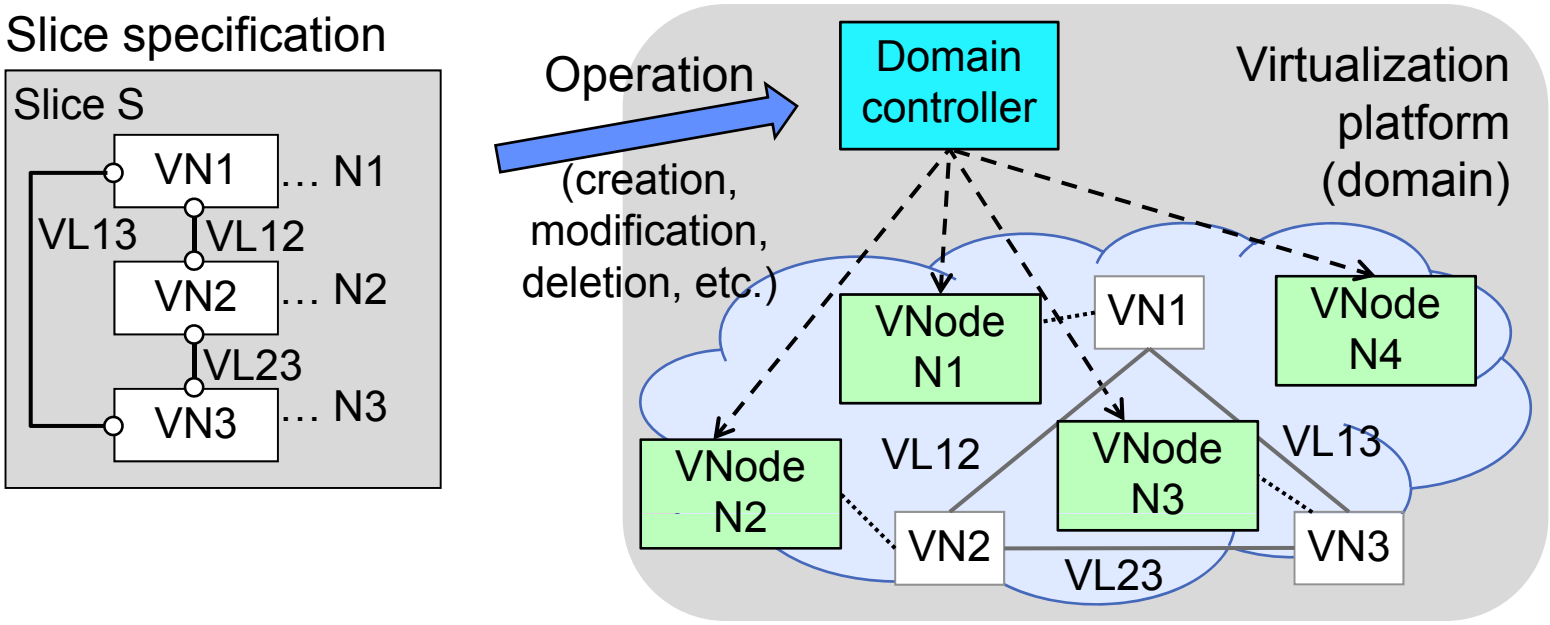
- **Multiple slices can be created on one physical network in this architecture and platform.**
  - Slices means virtual networks.



- **VNode (virtualization node) is a component of the network virtualization platform.**
  - VNode is a physical node.
  - VNode forwards packets on the platform as a router.
  - Slices are implemented as overlay networks on the platform.
  - VNodes are connected by tunnels using GRE/IP.
    - GRE (Generic Routing Encapsulation) is a protocol standardized by IETF.

Slice Creation and Management in the VNode Platform

- **The developer describes a slice design that represents the specification of the virtual network.**
  - The design describes virtual nodes, virtual links, and binds them (binding relationships between them).
  - The design is described by using an XML-based language.
- **A slice is created by sending a slice design to the domain controller (DC) of the VNode Platform.**



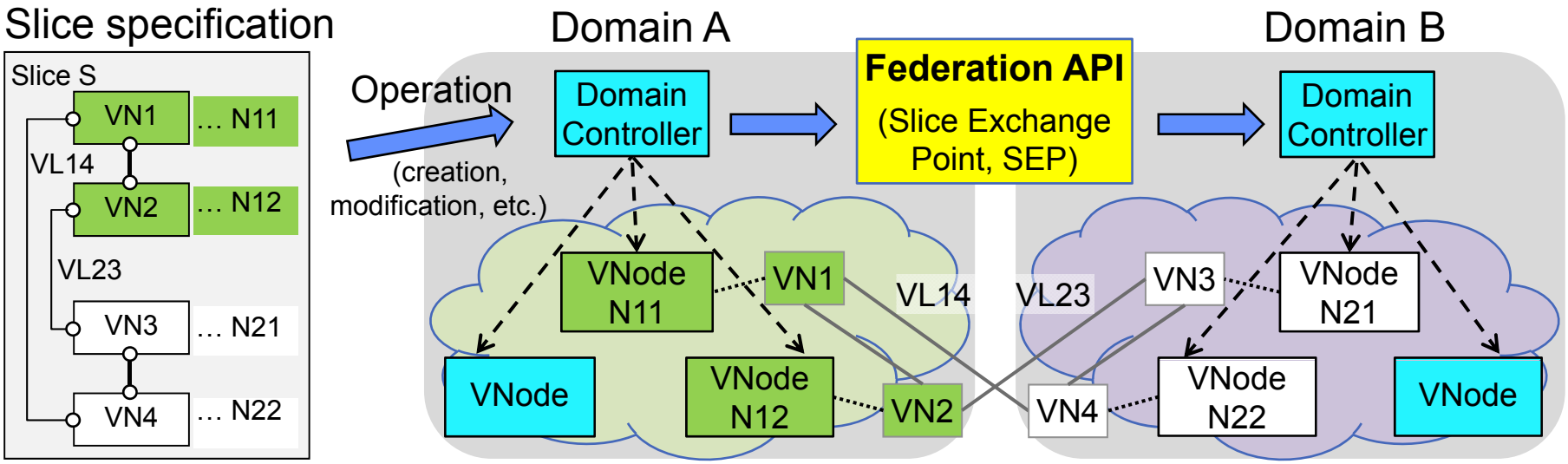
Federation between Virtualization Platforms

- **Federation functions between virtualization platforms**
  - Resource discovery functions
  - Slice handling functions
  - Queries on statistics and manifests
- **In this paper/presentation, we focus on *slice handling functions*, especially, *slice creation*.**
- **Slice handling functions**
  - Functions to create and to manage a slice that spread among multiple virtualization platforms from a single platform.

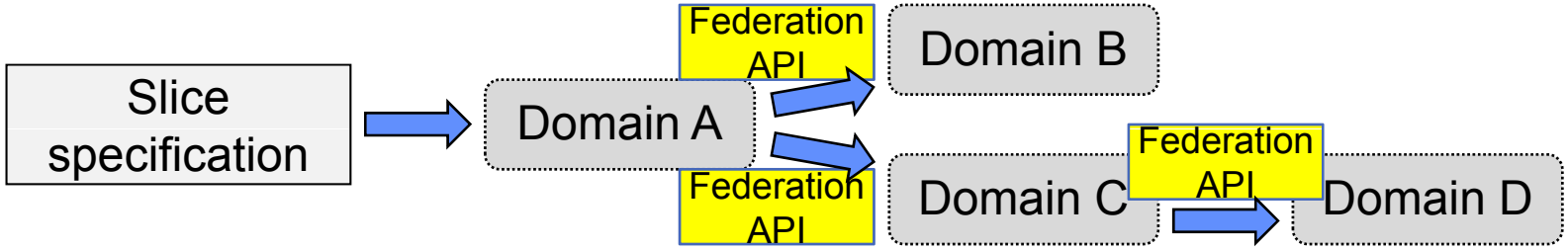
Federation between Virtualization Platforms (cont'd)

■ Basic federation method

◆ Instead of submitting the slice specification to both domains, it is passed to one domain and it passes the slice specification to the other domain.

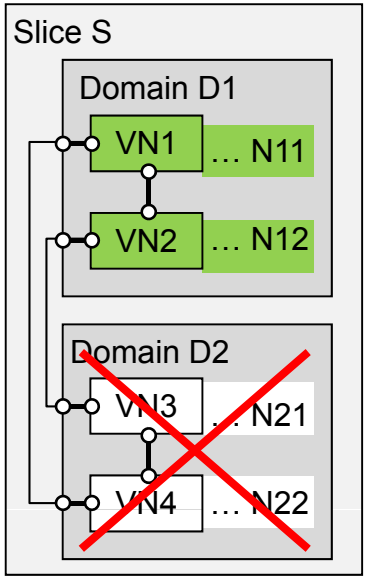


■ More complicated messaging pattern



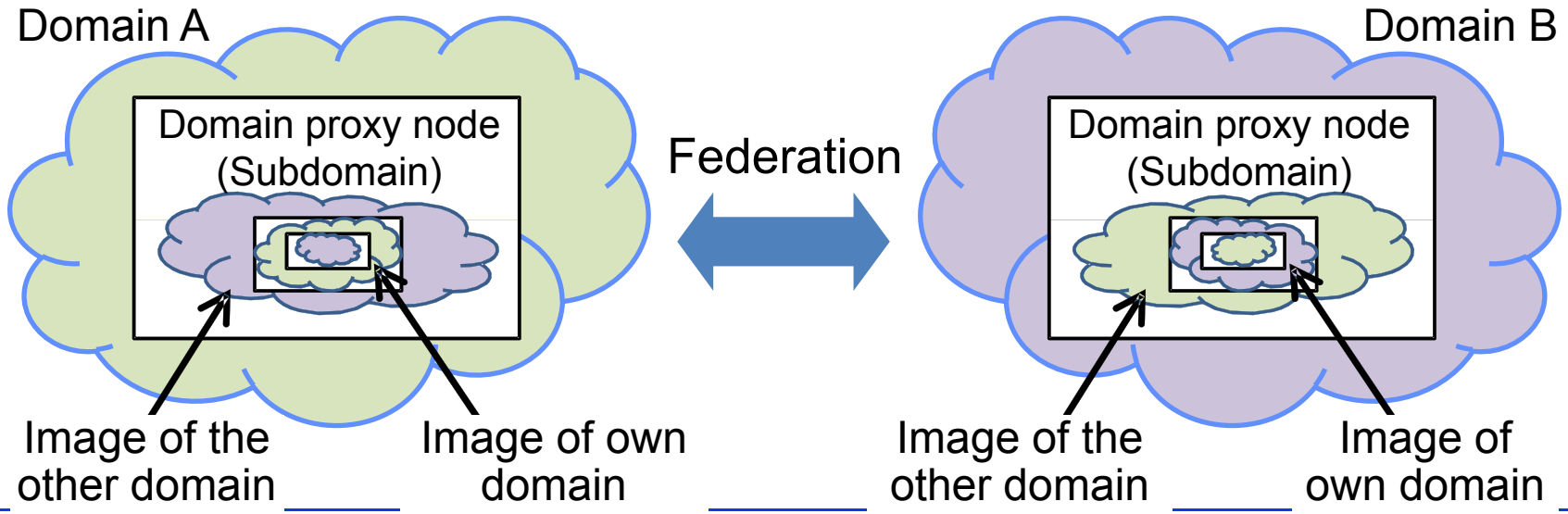
Conceptual Outline of Federation-less Federation (contribution)

- **A virtualization platform without federation functions does not have a concept of "other domain".**
  - The "own domain" is the only domain.
- **The part of the slice in other domain must belong to the own domain.**
  - This means that the other domain is a sub-domain of the own domain.
  - Information in the part of the slice in the other domain must be hidden from the DC in the original domain.
    - The part is to be managed only by the DC in the other domain.
    - Duplicated management of the part must be avoided, this part must be hidden.



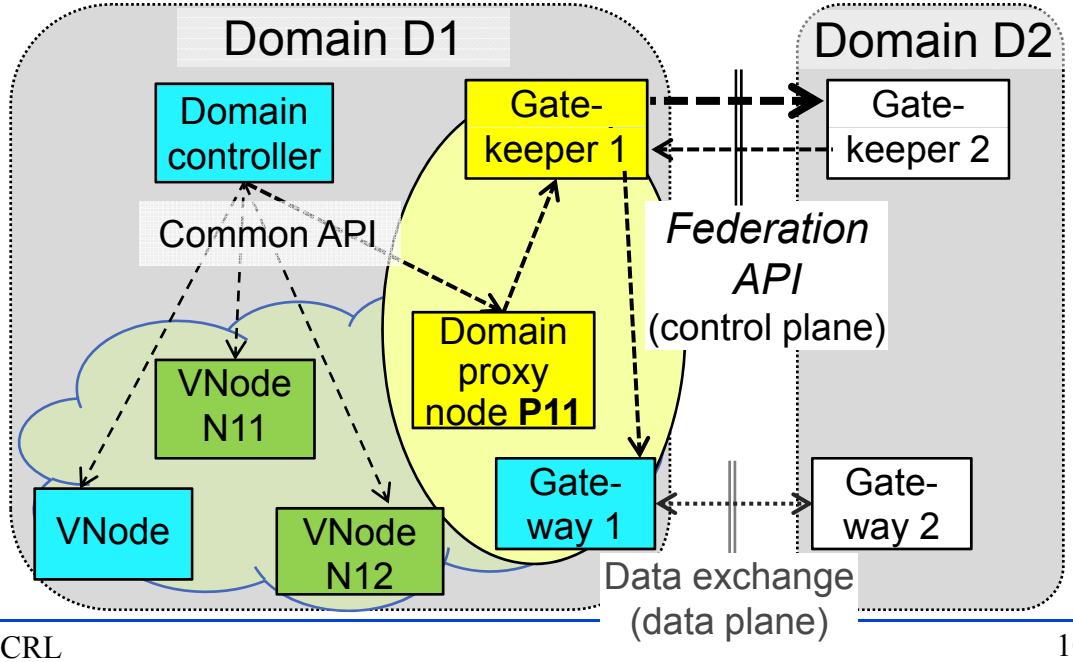
Conceptual Outline of Federation-less Federation (cont'd)

- **Domain proxy node (DPN) concept**
  - The only way to express a sub-domain is to use a virtual node.
    -- to express as a node belongs to a pseudo VNode (DPN)
  - DC can manage a DPN in the same way as a normal VNode.
- **DPN conceptually contains an image of the other domain.**
  - DC maps a *pseudo virtual node* (*PVN*) on a DPN by using the same way as for a normal virtual node.
  - The PVN must contain a part of the slice specification for the other domain (B) as a sub-structure.



Federation Architecture: Three components

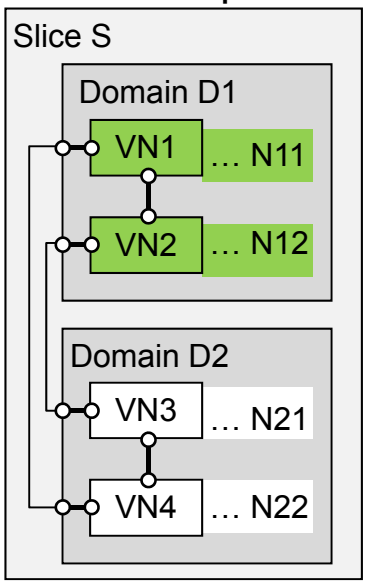
- **Domain proxy node (DPN)**
  - DPN receives a specification of PVN that contains the other-domain-part of slice definition and sends it to Gatekeeper.
- **Gatekeeper**
  - Gatekeeper receives the other-domain-part of slice definition and sends it to the other domain through the *Federation API*.
- **Gateway**
  - Gateway converts the data packet from intra-domain format to inter-domain format.
- **The numbers of DPN, gatekeeper, and gateway may be the same or different in federation of three or more domains.**



Homogenious federation
Process 0: Propositions

- **A symmetrical slice is used for example.**

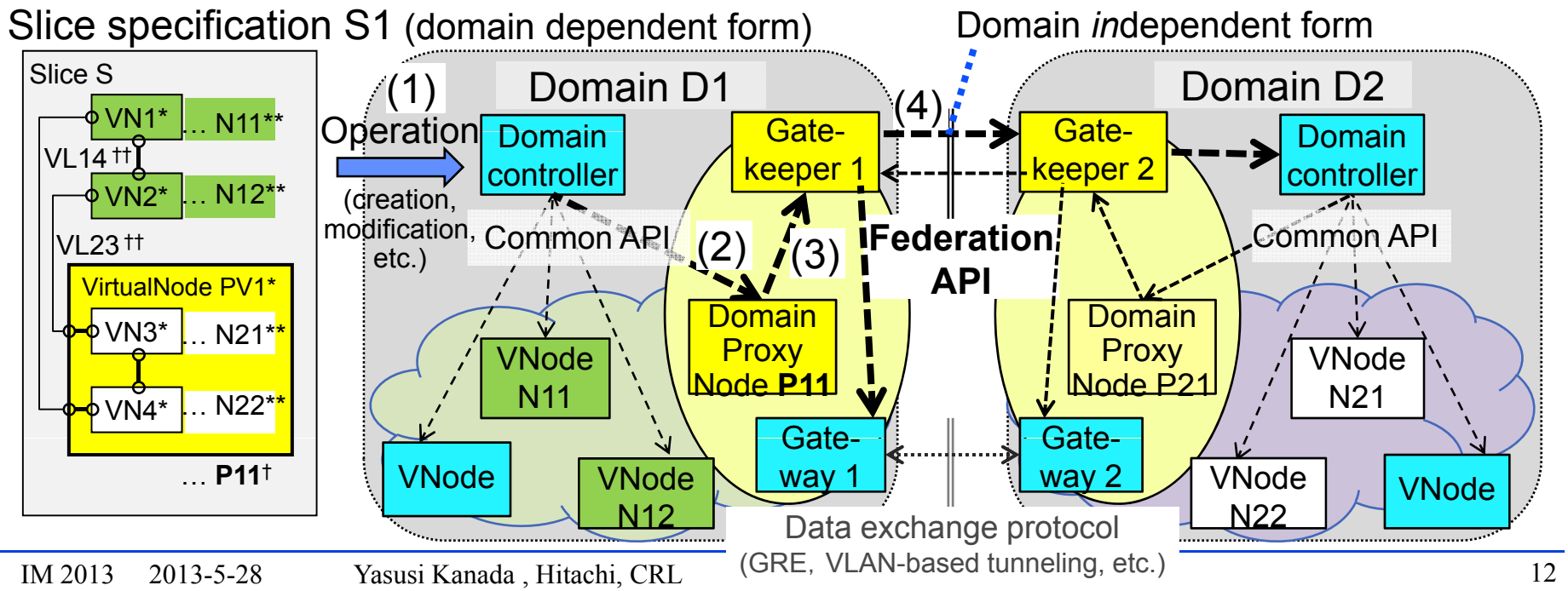
Canonical slice specification
(Domain-independent form)



- **This example handles a homogenious federation, but this method can be applied to heterogenious federations.**

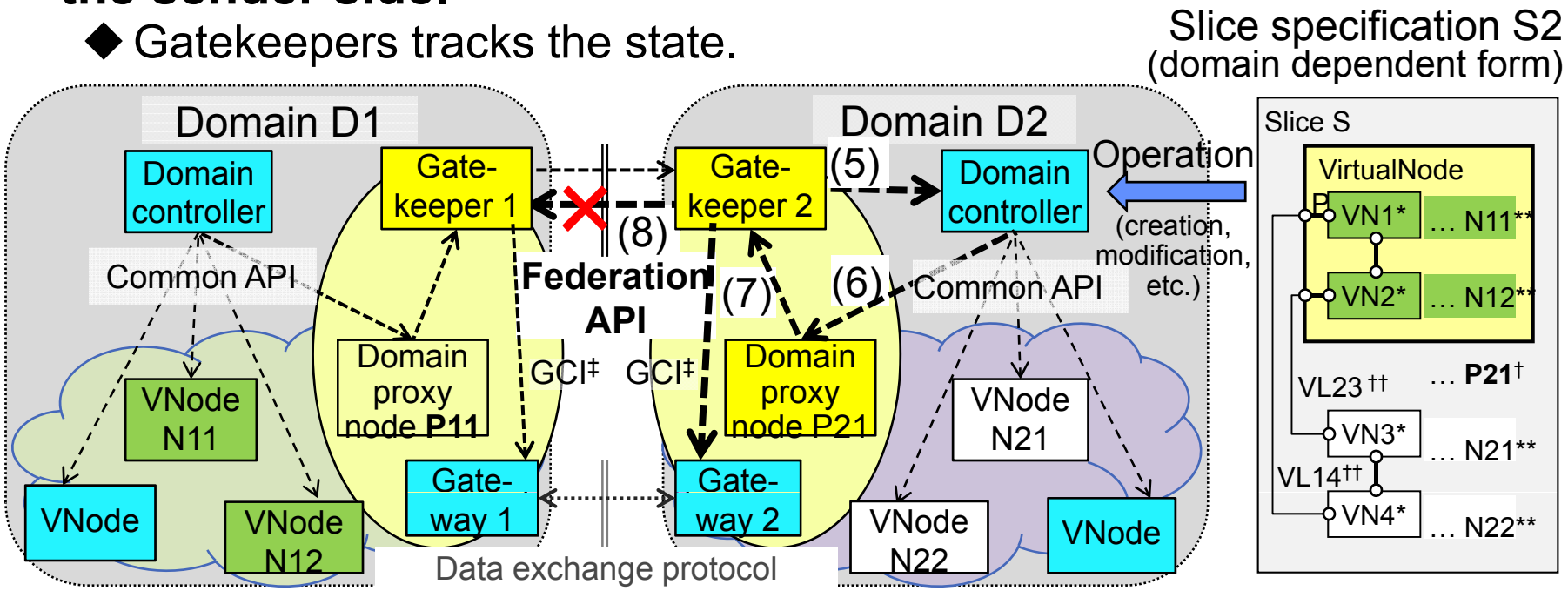
Homogenious Federation Process 1: Sender side

- **In a slice specification given to the DC, the PVN encloses the part of the slice for the other domain.**
  - The DC does not look at the inside of the PVN.
- **The DC distributes the slice specification to the DPN, as well as normal VNodes, using the same API** (step 2)**.**
- **The DPN sends the slice specification to the other domain through the gatekeepers** (steps 3, 4)**.**
  - The slice definition is in the domain-independent form.



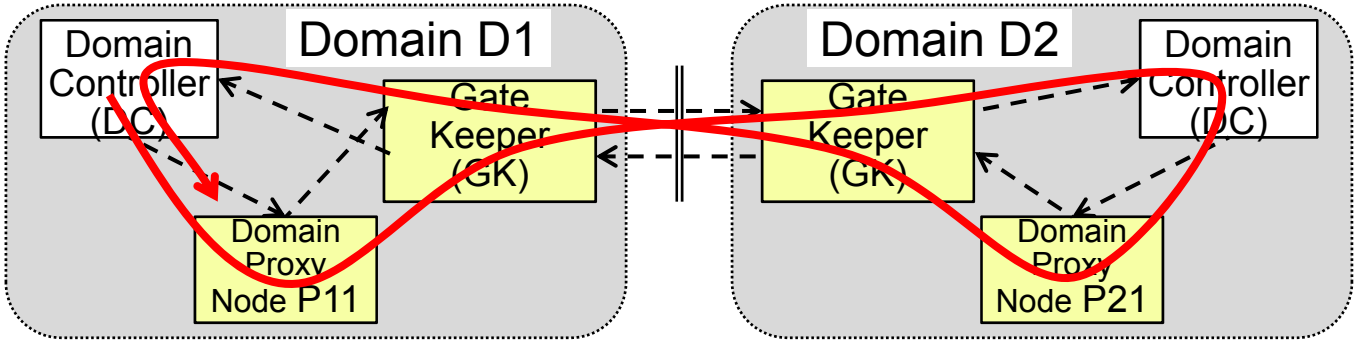
Homogenious Federation Process 2: Receiver side

- **The gatekeeper of the receiver-side encloses the part of the slice for the sender domain and send the slice specification to the DC** (step 5)**.**
- **The DC distributes the slice specification to the DPN, as well as normal VNodes, using the same API** (step 6)**.**
- **The slice definition is processed in the same method as in the sender side except the gatekeeper does not send it to the sender side.**
  - ◆ Gatekeepers tracks the state.



Message Loop Avoidance

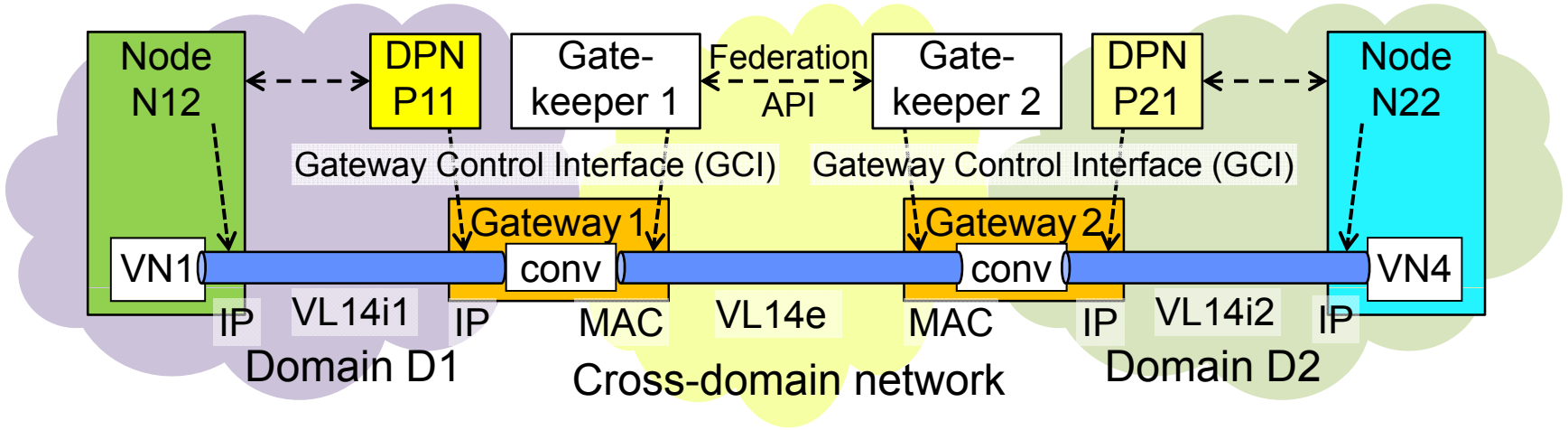
- **Inter-domain messages may cause an infinite loop because the conceptual structure is recursive.**
  - There may be many recursion patterns when there are three or more domains.



- **An Infinite loop must be avoided by using message identification and/or marking.**

Manageable Inter-domain Links for Non-IP Communication

- **A link that corresponds to each inter-domain virtual link is created by stitching three sections.**
  - The inter-domain section is created through the gateway control interface (GCI) while inter-domain messaging.
  - The intra-domain sections are created in the same method as normal virtual links in a domain.



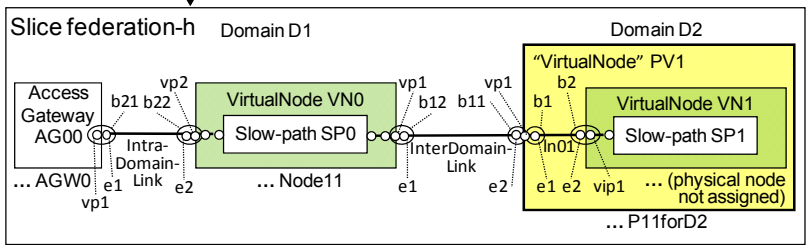
- **The gateways may convert inter- and intra-domain protocols if necessary.**
  - E.g., In the current VNode Platform, GRE is used for intra-domain, VLAN is used for inter-domain.
  - Gateways may be bipassed if no protocol conversion is required.

Issues in Federation-less-Federation Method

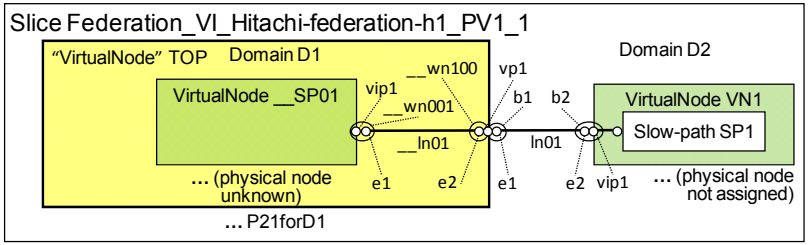
- **Restriction on modification**
  - If the domain does not have a command to update a virtual node, there is no way to update the structure of the other domain.
- **Difficulty in collecting information**
  - Resource discovery, statistical query, and asking manifests* may be difficult to implement because the DC of a domain does not collect information of the other domain.
    *manifests: virtual-node host-names or addresses, etc.

Implementation and Evaluation*

- **An example of slice specification used for the evaluation is shown.**
  - Slice specification given to domain 1
    - XML text →
    - Diagram ↓



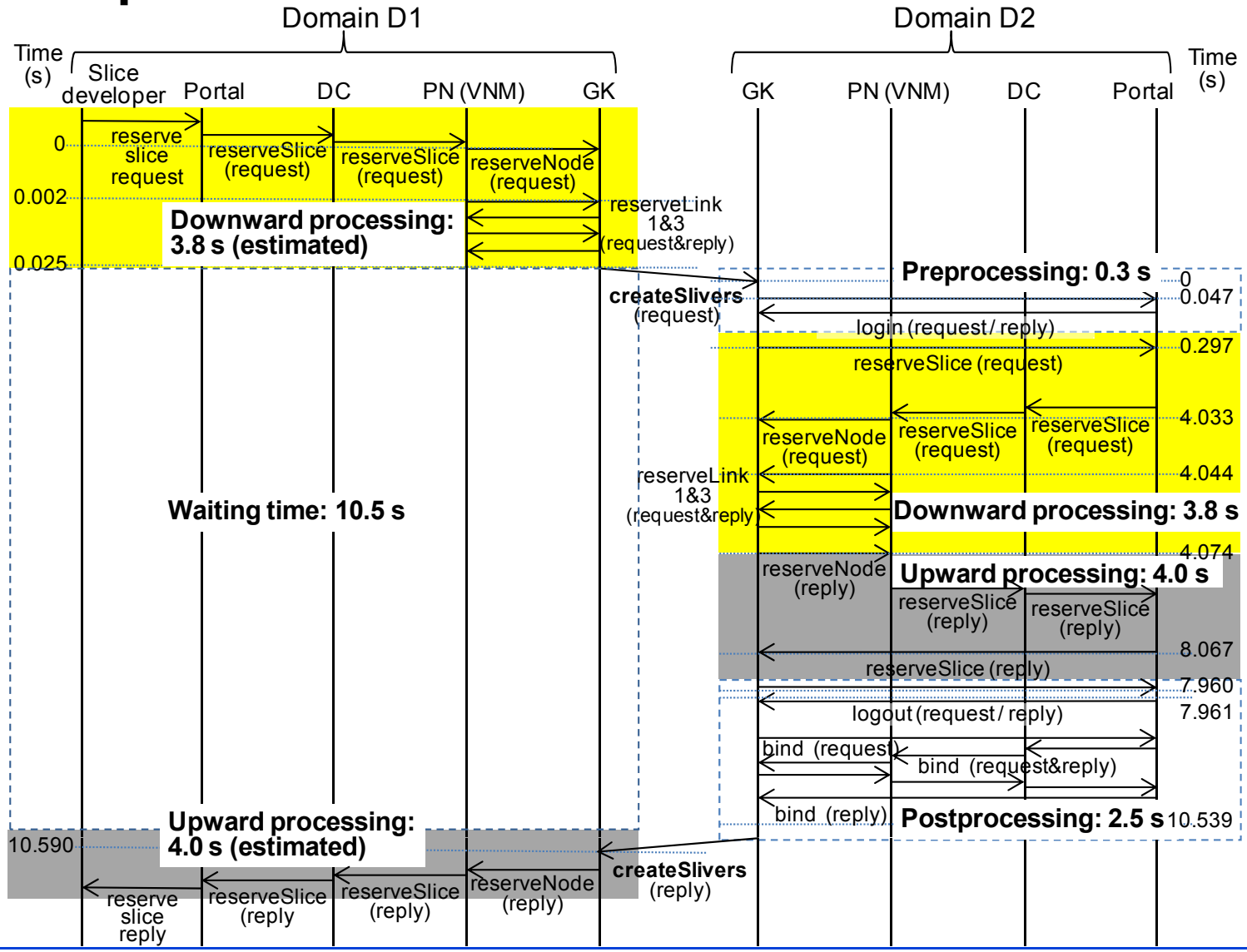
  - Slice specification generated for domain 2




```
<linkSliver name="InterDomainLink" type="link" subtype="GRE">
  <vports><vport name="e1" /><vport name="e2" /></vports>
</linkSliver>                                              InterDomain virtual link
<linkSliver name="IntraDomainLink" type="link" subtype="GRE">
  <vports><vport name="e1" /><vport name="e2" /></vports>
</linkSliver>                                              IntraDomain virtual link
</linkSlivers>
<nodeSlivers>
 <nodeSliver name="VN0" type="prog">
  <vports><vport name="vp1"/><vport name="vp2"/></vports>  Virtual node in domain D1
  <hierarchy>
   <sliverdef>
    <nodeSlivers>
     <nodeSliver name="SP0">                               A slow-path VM
      <vports><vport name="vip1" /><vport name="vip2"/></vports>
      <instance type="SlowPath_VM" subtype="KVM">
       <resources>...</resources>
       <params>
        <param key="bootImage" value="http://192.168.50.50/nict-test/sp/KVM_Ubuntu1010Server32.img" />
       </params>
      </instance>
     </nodeSliver>
    </nodeSlivers>
    <linkSlivers>...</linkSlivers>
   </sliverdef>
   <structure>...</structure>
   <params><param key="authKey" value="ssh-dss ... vnode@ubuntu1" /></params>
  </hierarchy>
 </nodeSliver>
 <nodeSliver name="PV1" type="prog">
  <vports><vport name="vp1"/></vports>                     Other domain (logical domain)
  <hierarchy>
   <sliverdef>
    <nodeSlivers>
     <nodeSliver name="VN1" type="prog">
      <vports><vport name="vip1" /></vports>               Virtual node in the other domain (D2)
      <hierarchy>
       <sliverdef>
        <nodeSlivers>
         <nodeSliver name="SP1">                           A slow-path VM
          <vports><vport name="vip1" /></vports>
          <instance type="SlowPath_VM" subtype="KVM">
           <resources>...</resources>
           <params>
            <param key="bootImage" value="http://192.168.50.58/nict-test/sp/KVM_Ubuntu1010Server32.img" />
           </params>
          </instance>
         </nodeSliver>
        </nodeSlivers>
        <linkSlivers>...</linkSlivers>
       </sliverdef>
       <structure>...</structure>
       <params><param key="authKey" value="ssh-dss ... vnode@ubuntu1" /></params>
      </hierarchy>
     </nodeSliver>
    </nodeSlivers>
    <linkSlivers>
     <linkSliver name="ln01" type="link"><vports><vport name="e1" /><vport name="e2" /></vports></linkSliver>
    </linkSlivers>
   </sliverdef>
   <structure>
    <bind name="b1"><vport sliverID="__TOP__" portname="vp1" /><vport sliverID="ln01" portname="e1" /></bind>
    <bind name="b2"><vport sliverID="VN1" portname="vip1" /><vport sliverID="ln01" portname="e2" /></bind>
   </structure>
   <params><param key="authKey" value="ssh-dss ... vnode@ubuntu1" /></params>
  </hierarchy>
 </nodeSliver>
 <nodeSliver name="AG00" type="agw"><vports><vport name="vp1"/></vports></nodeSliver>
</nodeSlivers>                                             Virtual access gateway
</sliverdef>
<structure>
 <bind name="b11"><vport sliverID="PV1" portname="vp1" /><vport sliverID="InterDomainLink" portname="e1" /></bind>
 <bind name="b12"><vport sliverID="NS00" portname="vp1" /><vport sliverID="InterDomainLink" portname="e2" /></bind>
 <bind name="b21"><vport sliverID="AG00" portname="vp1" /><vport sliverID="IntraDomainLink" portname="e1" /></bind>
 <bind name="b22"><vport sliverID="NS00" portname="vp2" /><vport sliverID="IntraDomainLink" portname="e2" /></bind>
</structure>
</slicespec>
<mapping slice="federation-h1" vnetwork="Slice">
 <amap node="PV1" vnode="P11forD2" />                      Logical-physical domain mapping
 <amap node="VN0" vnode="Node11" />
 <amap node="AG00" vnode="AGW0" />                         Logical-physical node mapping in own domain
</mapping>
</slice-design>
```

Implementation and Evaluation (cont'd)*

- The sequence and measured time is as follows.



Conclusion

- **This paper proposes a method of federation between domains without a federation function.**
- **The proposed method enables non-IP data communication on the slice.**
- **The federation method was successfully implemented on the VNode Platform.**
- **Future work includes heterogeneous federation, especially federation between VNode Platform and ProtoGENI.**
  - A limited implementation has been already demonstrated in GEC 16 (i.e., 16th GENI Engineering Conference).