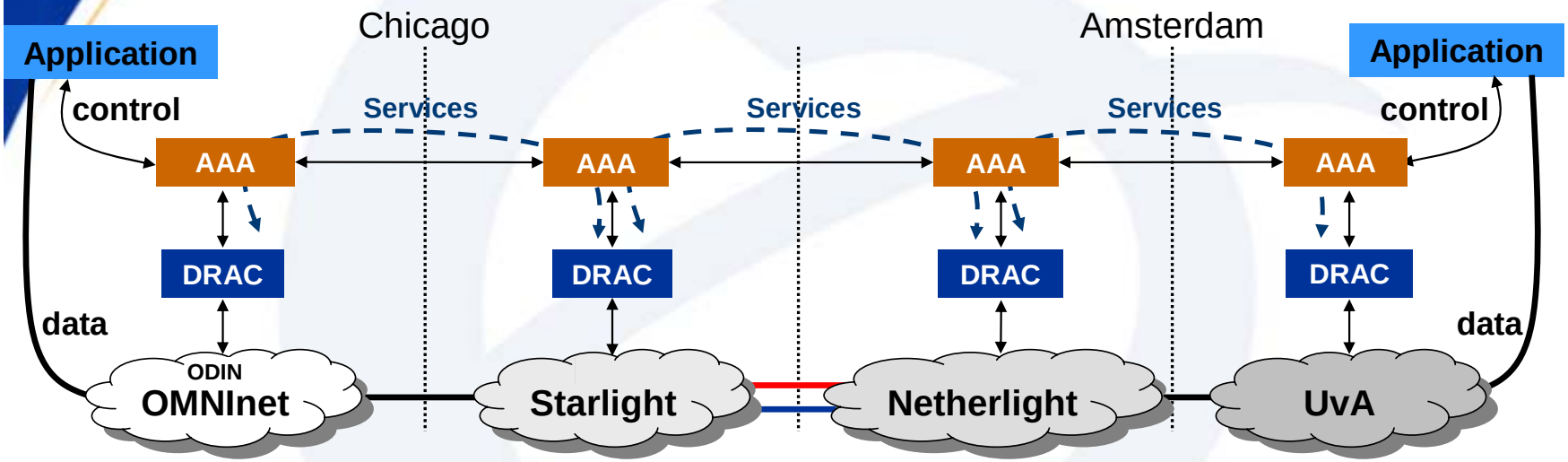
Phosphorus and DICE IDC: Multi-domain Project approaches

Bram Peeters, Surfnet
Inder Monga, Nortel

Multi-domain – Brief History

SC2004 CONTROL CHALLENGE



- finesse the control of bandwidth across multiple domains
- while exploiting scalability and intra- , inter-domain fault recovery
- thru layering of a novel SOA upon legacy control planes and NEs

Today

- Many intra-domain implementations
  - DRAC, UCLPv2, DRAGON, AutoBAHN, VIOLA..
- Few multi-domain projects
  - Phosphorus, Oscars/DICE IDC, AutoBAHN IDM
- Focus of the talk today
  - Brief overview of Phosphorus and DICE IDC
  - Raise discussion points

Phosphorus Aim and Drivers

- Create transparent *network service for middleware*

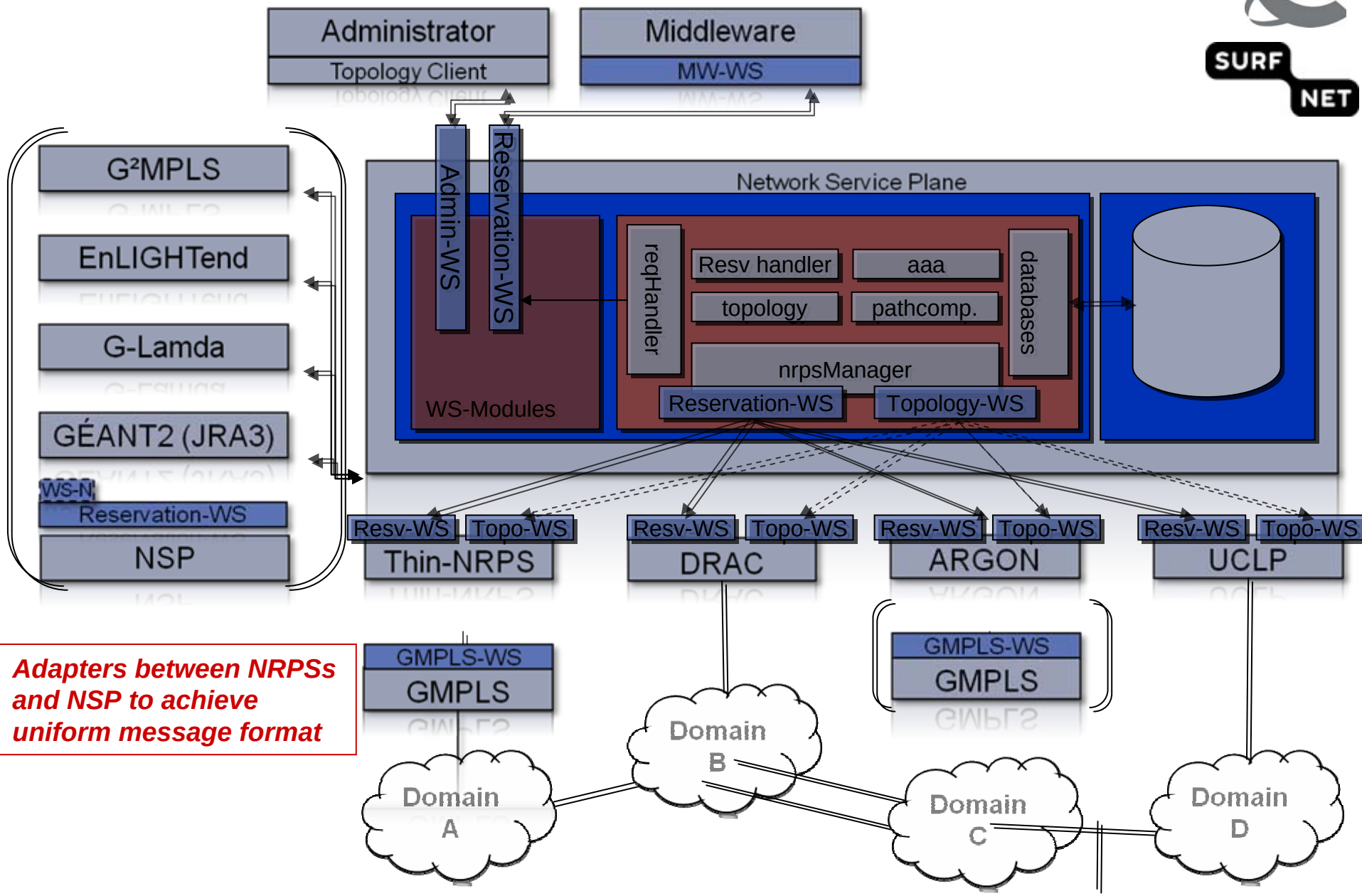
- WP1 focus: Integrate **existing heterogeneous** ‘domain controllers’
  - Aim is to provision circuits seamlessly across several domains
  - Controllers in the project: UCLP, Viola/ARGON, DRAC
    - Assumed to minimally provide WS-based (circuit) scheduling and topology capabilities

- Work towards <u>prototype within first year of project</u>
  - Quick integration => KISS, and make it work
  - Chosen to work with ‘multiple centralized intelligence’ approach
    - Domain controller has all the domain knowledge, NSP is just a client…
      - Broker principle!!!
    - Centralized intelligence in a single NSP removes a lot of problems

Terminology

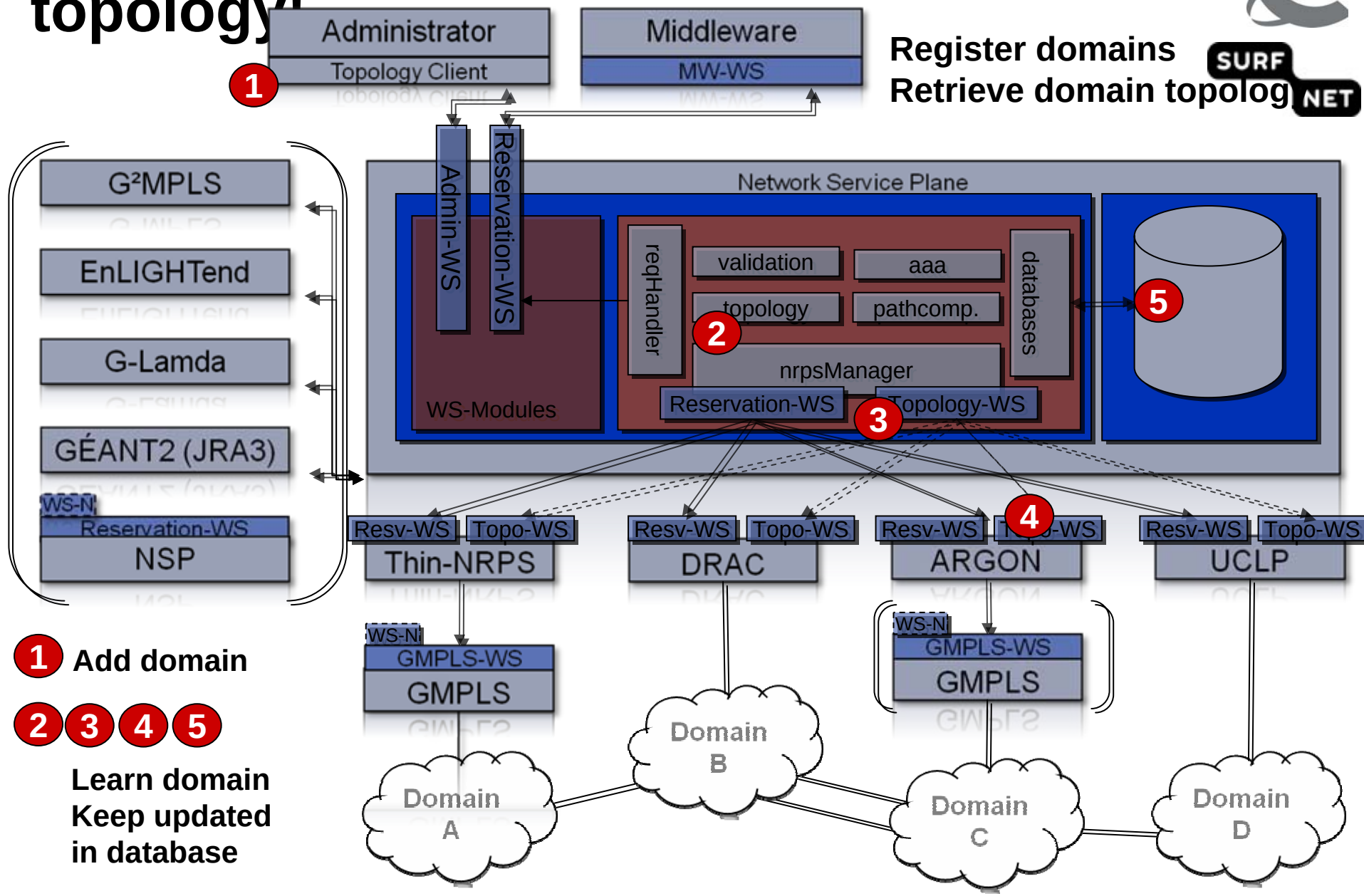
- NRPS = Network Resource Provisioning System
  - Local domain controller providing the service on a single domain
- NSP = Network Service Plane
  - Global broker creating the E2E service using the different NRPS

Architecture :: overview

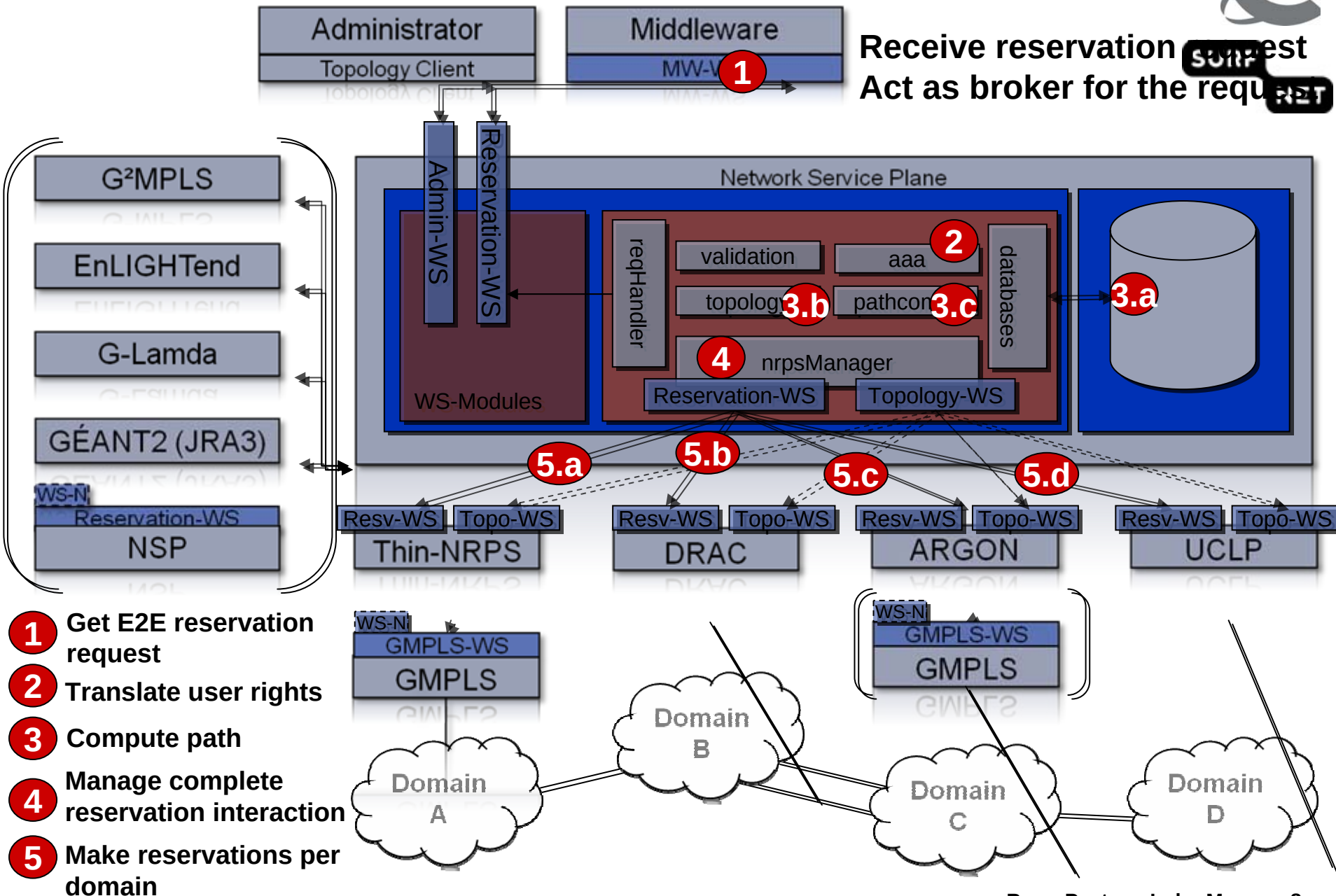


Adapters between NRPSs and NSP to achieve uniform message format

architecture :: framework flow – topology



architecture :: reservation flow



Architecture overview: reservation interfaces

No extension of reservation capabilities of single domain

- Basic function: reserve between two endpoints on a domain (Client or Border – or “UNI, ENNI”)

- Pre-reserve + commit
- Cancel
- Notifications

- Scheduling, channelisation, and technology stitching coordinated by NSP, but works on info from NRPSes
  - Remember – NSP is only ‘a’ broker

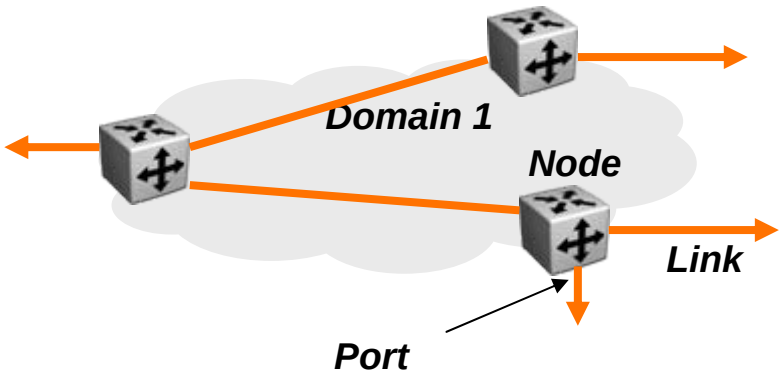
Architecture overview: topology interfaces

- Learn “topology” of a domain
  - Topology = List of endpoints with links between them => abstracted to Single-Node-Domain or full mesh links
  - Retains possibility of more complex topologies

- Links between domains administrated per domain
  - Information checked for consistency by NSP
  - Knowledge is local anyway, all technologies ‘covered’ for discovery
  - Need to agree global ‘cost’?

- This topology information can be given out to multiple NSPs

DICE Topology

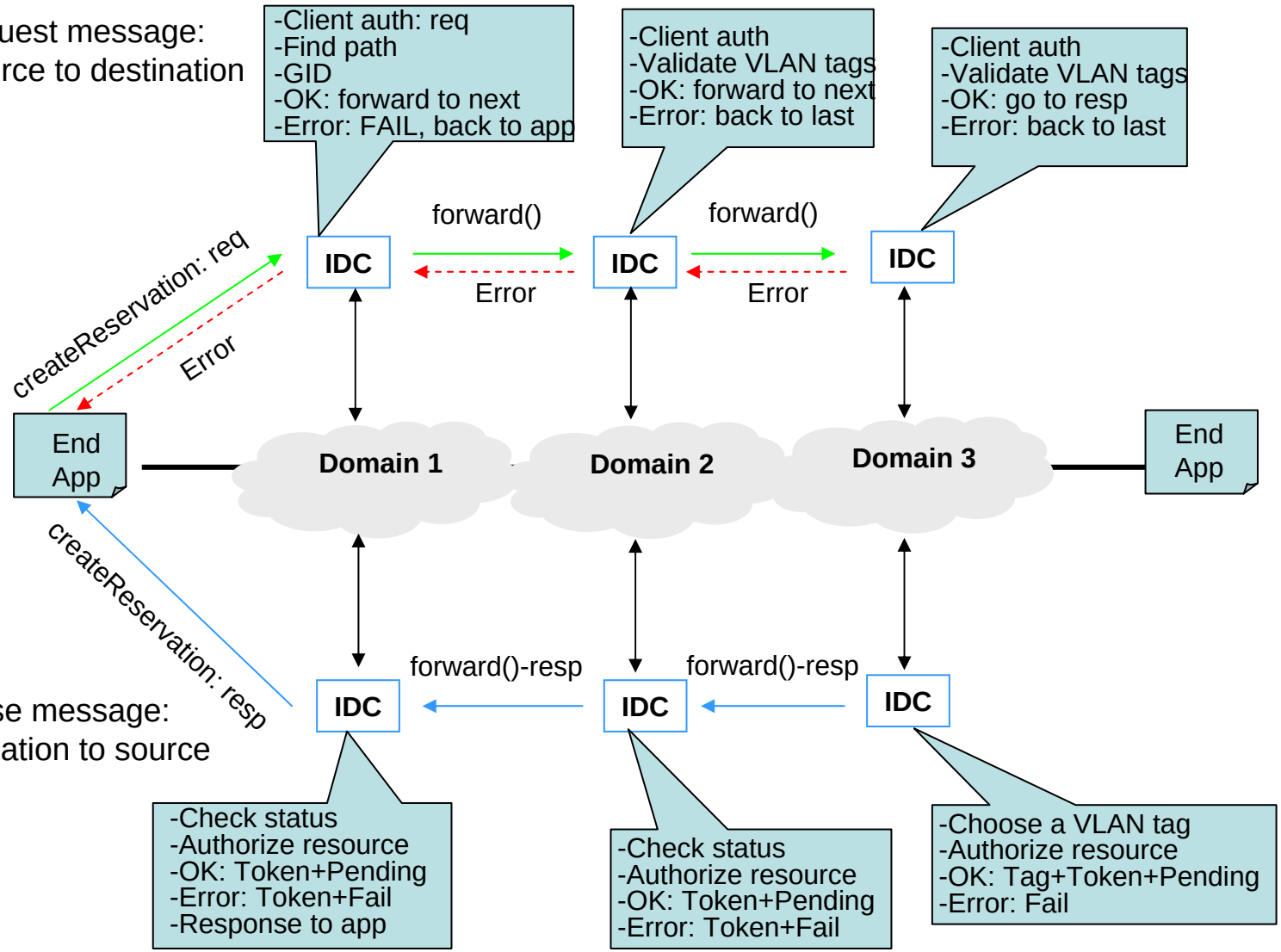


- Topology Hierarchy
  - Domain
  - Node
  - Port
  - Link

- Port and link capabilities as well as available capacity can be specified as well
- RemoteLink descriptors require manual configuration during inter-domain startup
- Accounts for both push and pull models
  - First implementation supports the simpler pull model

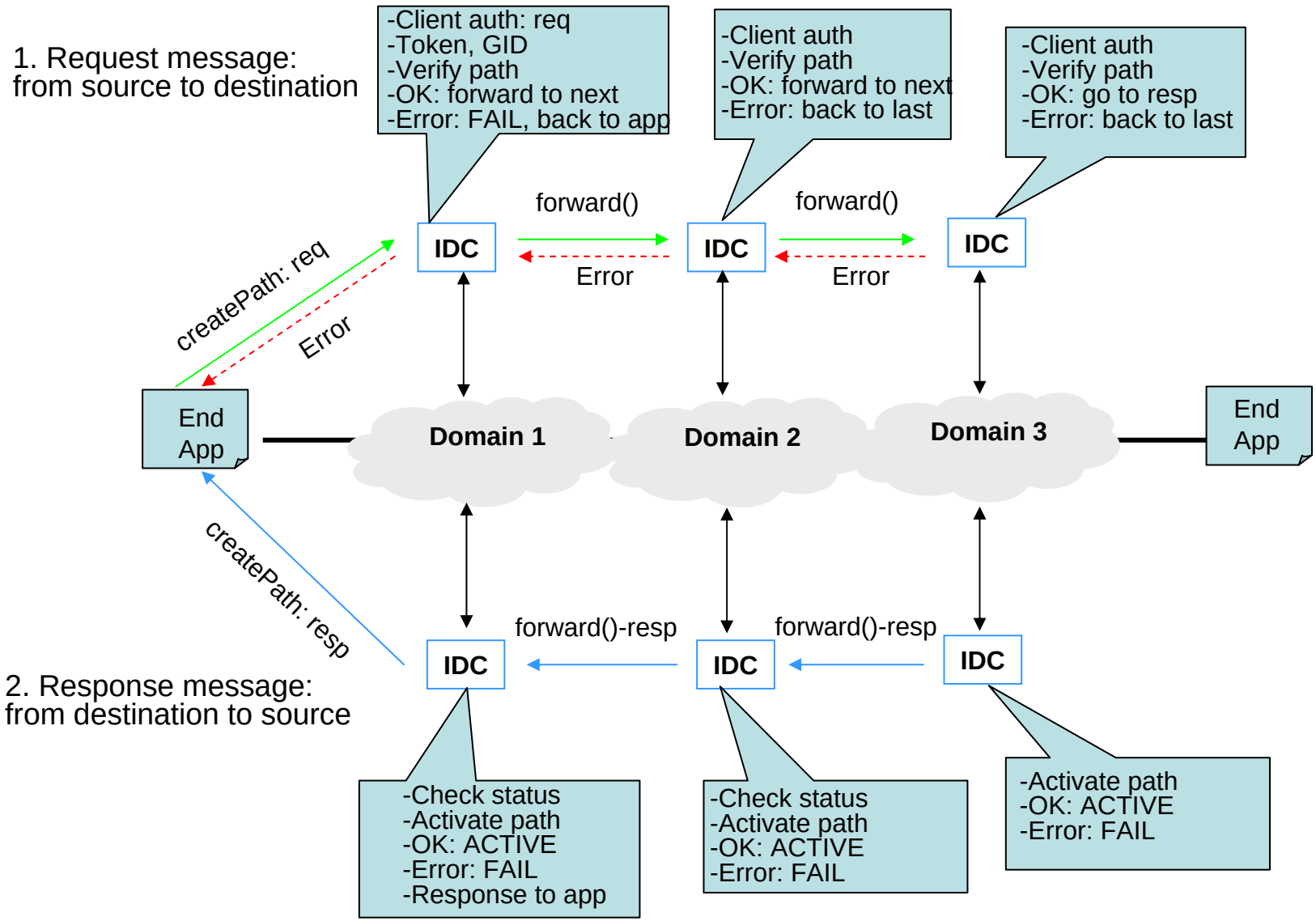
DICE IDC Overview: Resource Scheduling example

1. Request message: from source to destination

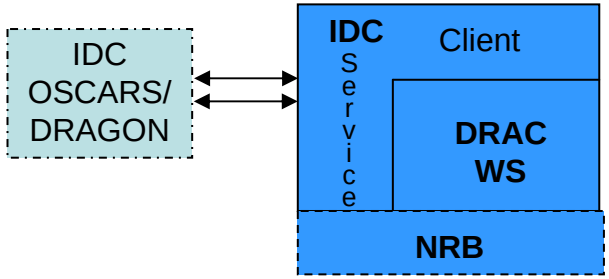


2. Response message: from destination to source

DICE IDC Overview
Resource Signaling Example



Implementing DICE on DRAC



DICE IDC

- Topology: IDC
  - getNetworkTopology
  - initiateTopologyPull
- Reservation: User
  - createReservation
  - cancelReservation
- Signaling: User
  - createPath
  - refreshPath
  - teardownPath
- Relay on reservation and signaling: IDC
  - forward
- Monitoring: User
  - queryReservation
  - listReservations

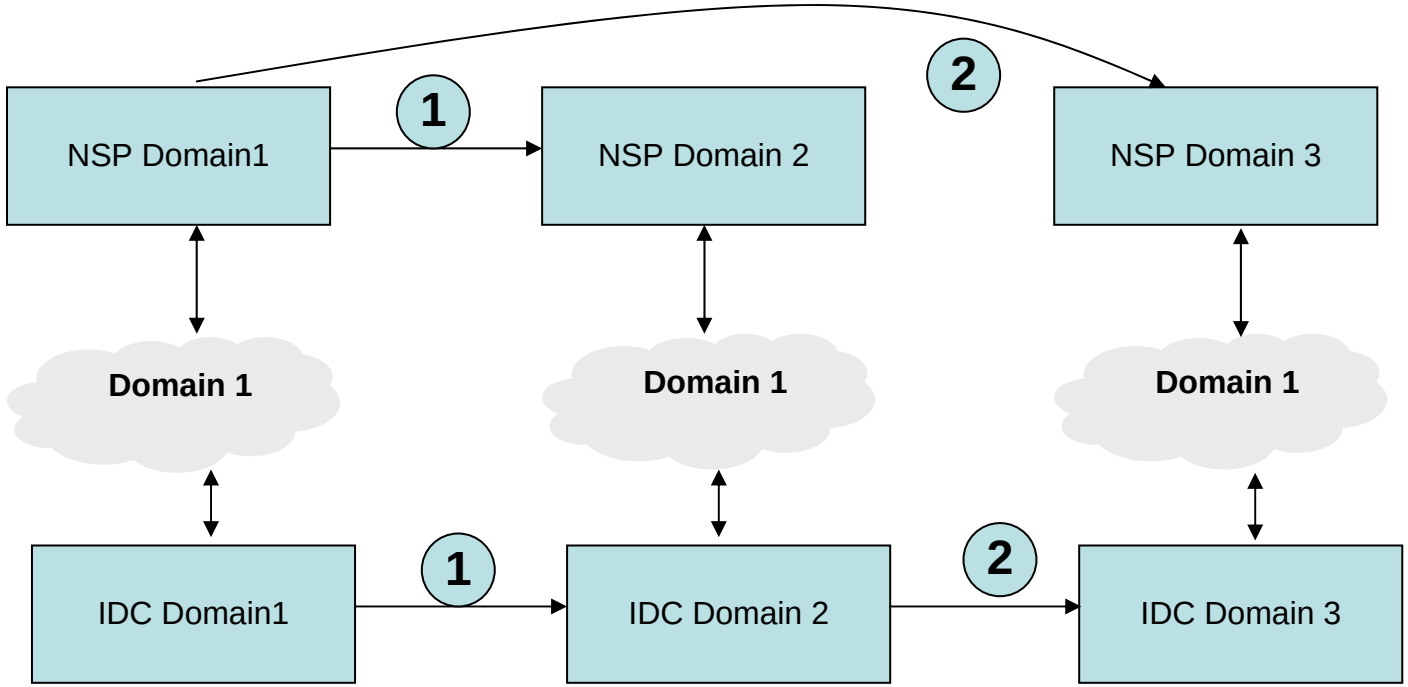
DRAC

- Topology (in network monitoring)
  - queryEndpoints()
- Resource Scheduling
  - authenticate
  - createReservationSchedule
  - cancelReservationSchedule
  - addReservationOccurrence
  - cancelReservationOccurrence
- Signaling
  - activateReservationOccurrence
  - confirmReservationSchedule
- IDC
  - Same interface commands
- Monitoring
  - queryPathAvailability
  - queryReservationSchedules
  - queryReservationOccurrences
  - queryReservationOccurrenceAlarms

Comparing trajectories: Raising GLIF Discussion points

- Architectural model
  - Centralized vs Distributed
  - Hybrid?
- Topology exchange
  - Most challenging, define the scope
- Path Computation
  - First domain vs Hop by Hop
  - Availability/Utilization
- Signaling
  - WS versus protocols
- Path Management
  - Refresh Path State

Architectural Model

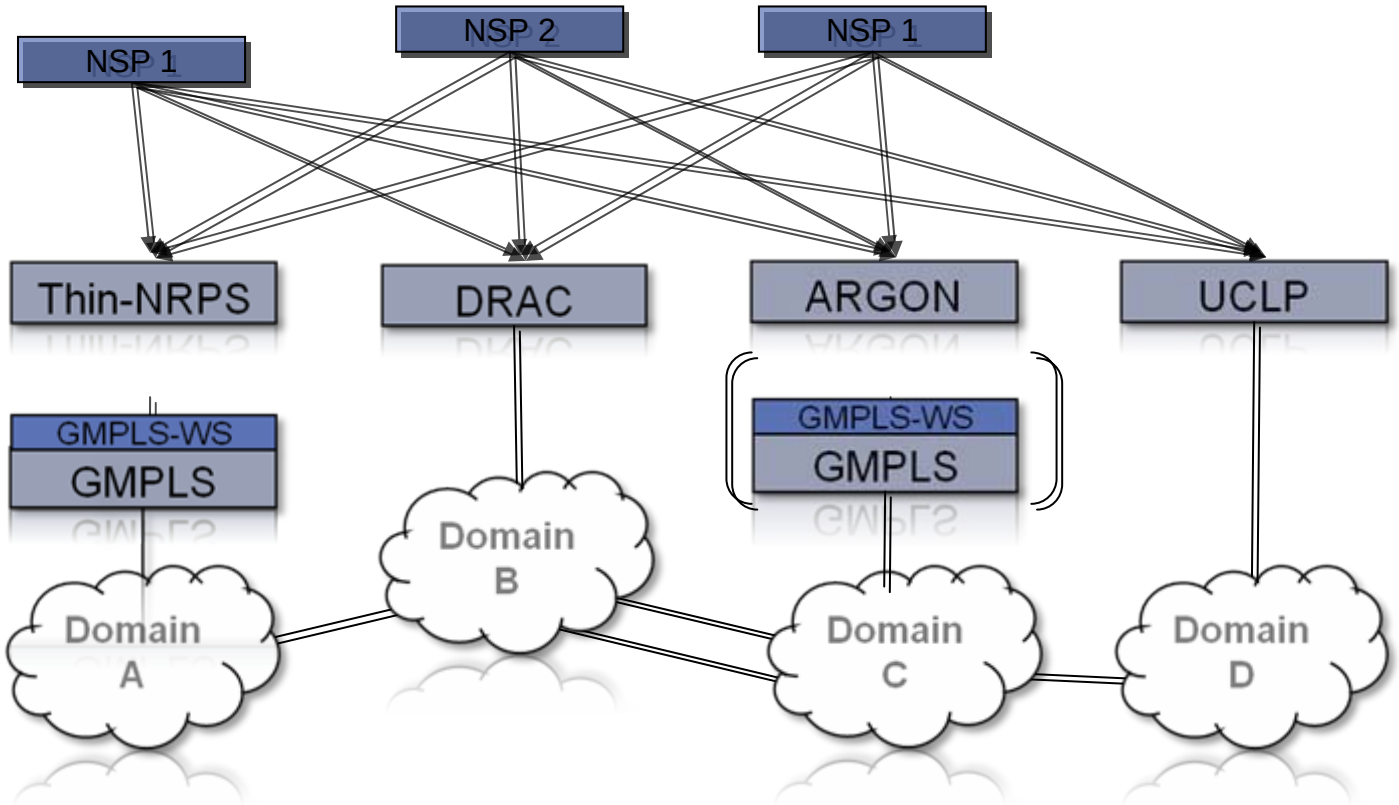


- Centralized/Tree model more suited towards co-scheduling of multiple resource types (multiple point to points)
  - Applications like Grid-computing
  - “Optimization” with longer term views possible
- Distributed suited for point to point dynamic services
- Both models support the flexibility though initially
  - Phosphorus implements the tree model
  - IDC implements the distributed model

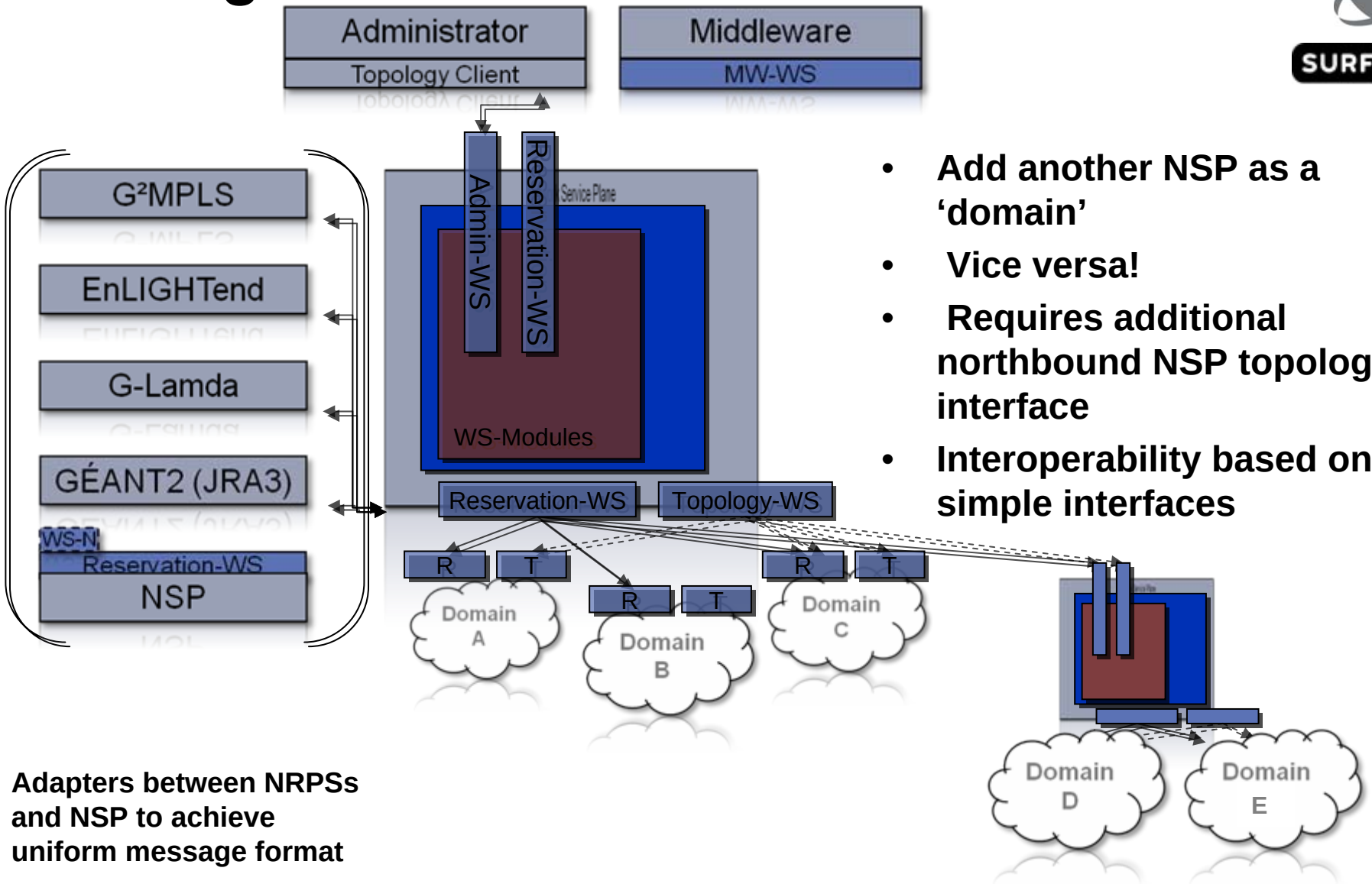
Scaling 1: multiple NSPs

-NSPs are clients of the NRPS
-A single NSP can do the E2E provisioning

➔ Every domain can have its own, independent NSP



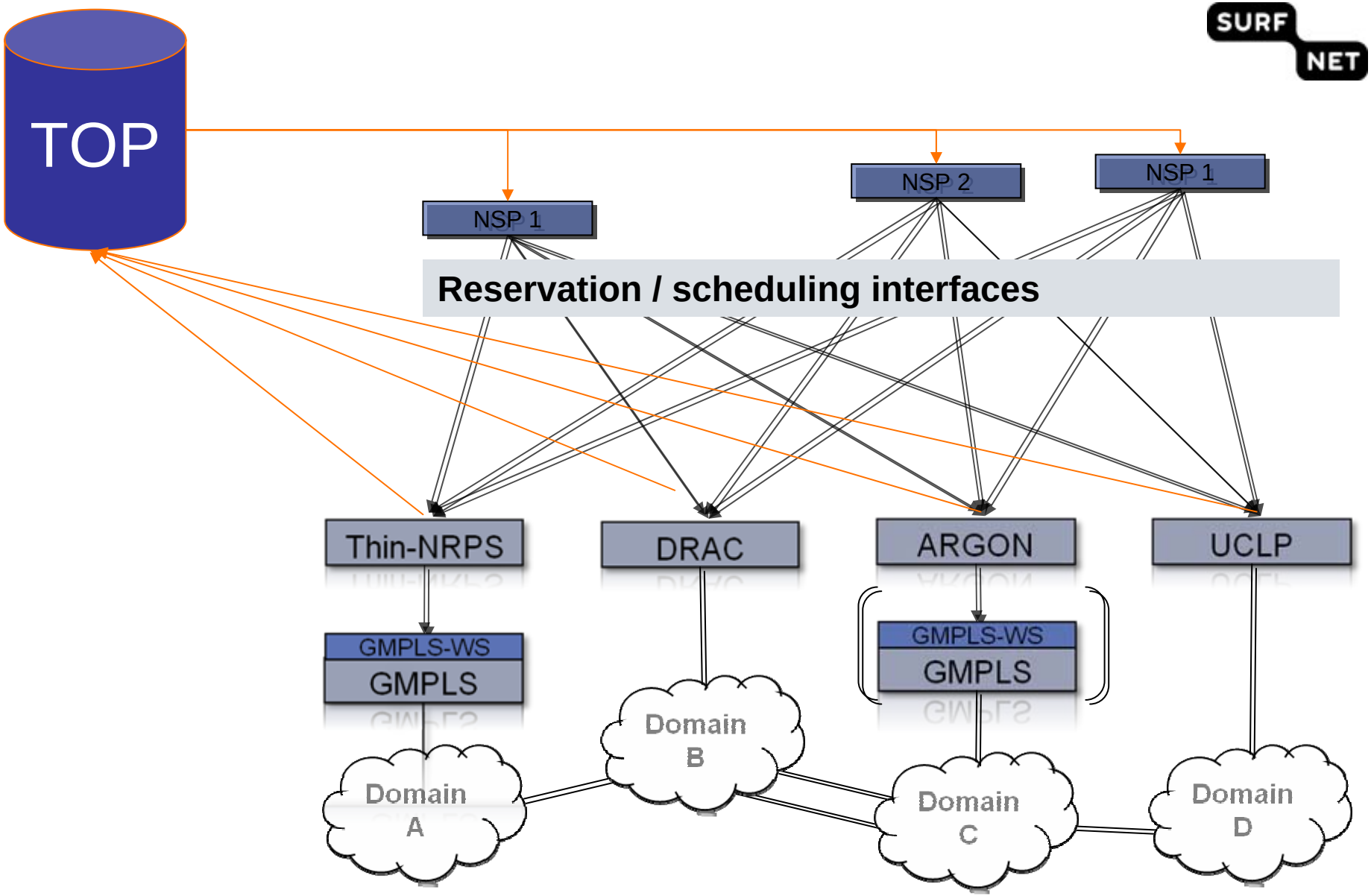
Scaling 2: hierarchical NSPs??



- Add another NSP as a 'domain'
- Vice versa!
- Requires additional northbound NSP topology interface
- Interoperability based on simple interfaces

Adapters between NRPSs and NSP to achieve uniform message format

architecture :: scaling: treat topology separately



Topology and Path Computation (contd.)

- Represents some of biggest challenges
  - What topology distribution mechanism?
    - Do you trust neighboring domains?
    - How often is the topology exchange (hrs vs mts vs secs)
  - Trial-and-error probabilities seem high
    - Calculating IDC does not have any idea of schedules, just a snapshot of “near-current” utilization
    - Failures do not necessarily increase the probability of success
  - Inter-domain restoration model not addressed
    - Which domain is responsible for restoration?
    - How do you identify path failure so proper restoration steps can be initiated?
  - How to “discover” inter-domain connections
    - Manual configuration for domain edges
- Research areas
  - Publish summarized schedules along with current utilization
  - Ability to specify unconstrained paths (no start/end time, just total time)
  - Multi-layer

Path Computation

- The IDC's currently compute the desired path before reservation
- How about using IP forwarding methods for path computation?
  - Just go to next hop domain
  - Let each domain IDC determine which next domain to forward the signal to
  - Retries handled by intermediary domain adjacent to failure

Signaling protocol

- Web Services vs GMPLS signaling
  - Phosphorus standardizes on Web Services
  - DICE IDC supports both options
    - RSVP-TE or Web Services or both
- Should we standardize on one versus the other?
  - Policy enforcement at the inter-domain edge is a MUST
    - Both business and network policies
  - Flexibility of policy enforcement and richness of information exchange easier at the NSP/IDC layer
  - Propose we converge on WS for inter-domain negotiation
    - Intra-domain can be either TL1, CLI, GMPLS, WS, etc.

Path Management

- Path refreshes (inline or through WS) are proposed by DICE IDC
  - Seems like overkill to me
- Phosphorus brings down path based on schedule or end-user request
- Path failure detection mechanisms needed
  - Which domain?

Conclusions

- High degree of correlation in the diverse multi-domain projects
  - Phosphorus, IDC, Autobahn, G-L/EL, DRAC/UvA
- Apply multi-domain topology abstraction concepts to the GOLE
- What next?
  - Share research experiences and agree on topology exchange, reserve messages and path compute model
  - Converge on an IDC implementation for a longer term “live” research testbed
  - Tackle failure models and restoration

THANK YOU!

For more information:

bram.peeters at surfnet.nl

imonga at nortel.com