# StorNet

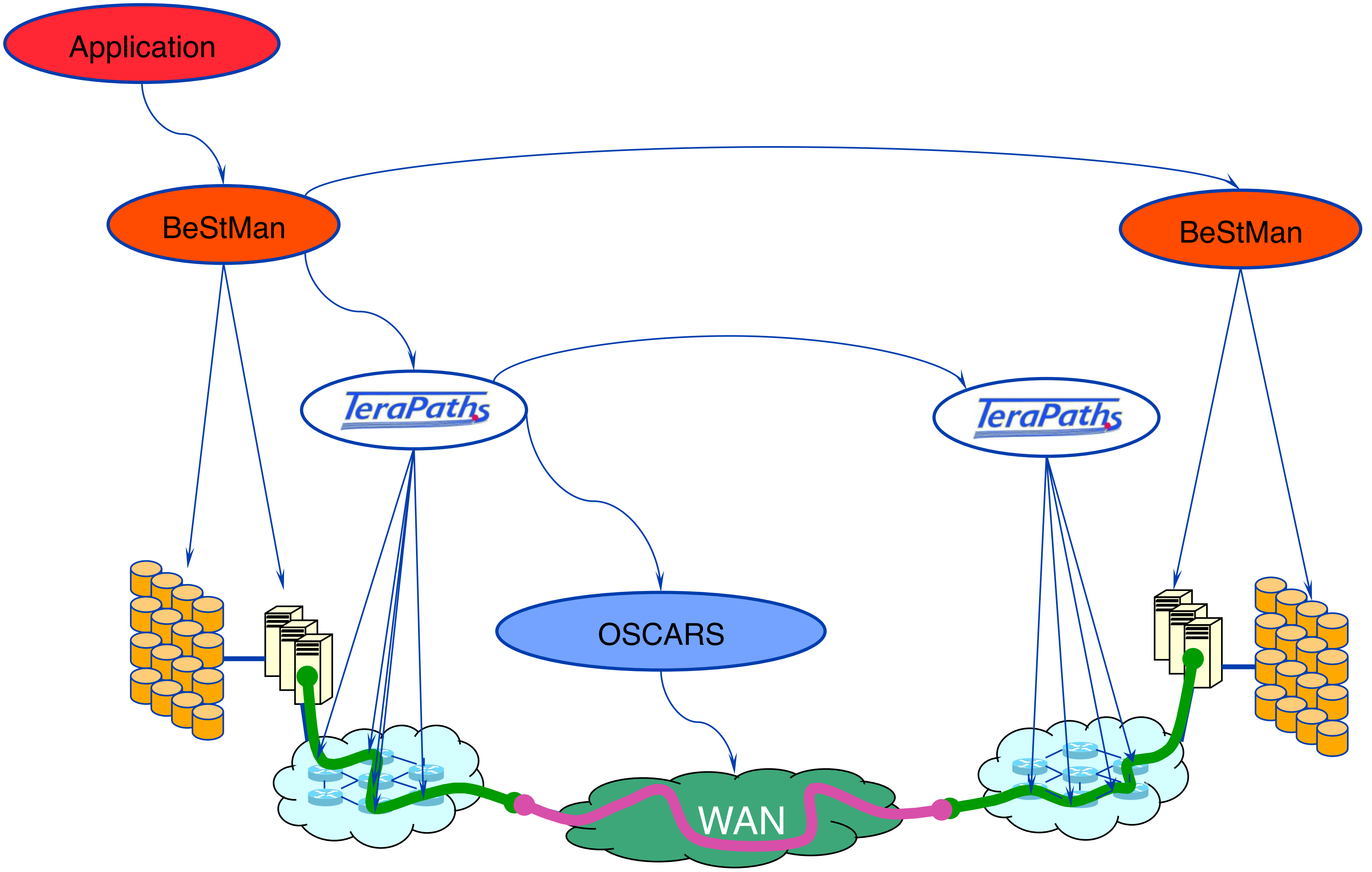
## Integrated Dynamic Storage and Network Resource Provisioning and Management for Automated Data Transfers

**Alex Sim**

**In collaboration with**

**BNL: Dimitrios Katramatos, Dantong Yu**
**LBNL: Junmin Gu, Vijaya Natarajan, Arie Shoshani, Alex Sim**
**U. Michigan: Shawn McKee**

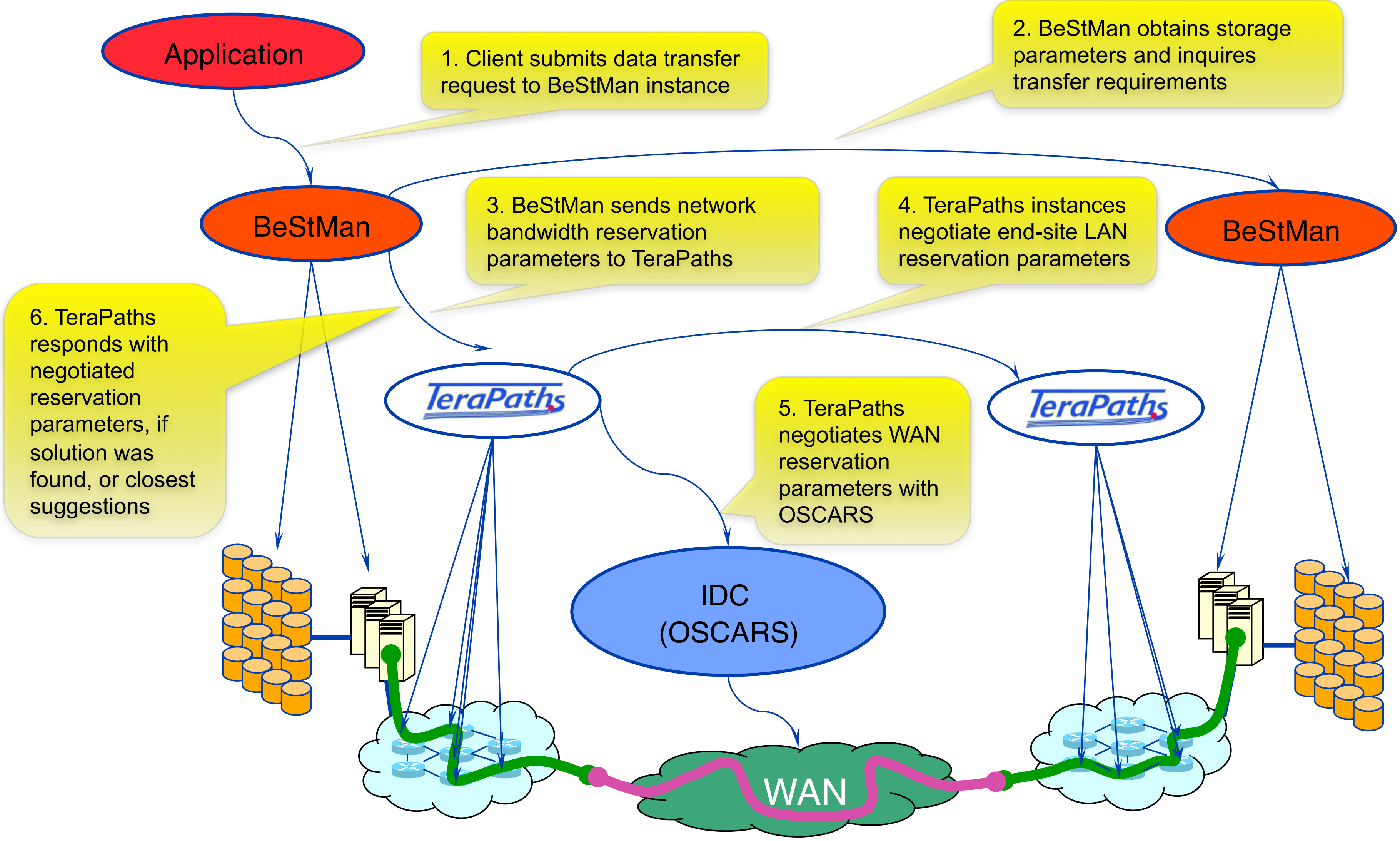
# StorNet


Application
BeStMan
BeStMan
TeraPaths
TeraPaths
OSCARS
WAN

# StorNet Project Overview

- **Project Goals:**
  - Design and develop an integrated end-to-end resource provisioning system for high performance data transfers
  - Improve resource utilization by co-scheduling network and storage resources and ensure data transfer efficiency
  - Support end-to-end data transfers with a negotiated transfer completion timeline.
- **Impact of StorNet on Science**
  - Scheduling Network and Storage as a 1st Class Resource through Virtualization
  - Provide a holistic approach for DOE data-intensive applications to share data
  - Provide data management capabilities commensurate with exascale computing

# StorNet: System Design and Implementation
# Integration of TeraPaths with BeStMan (SRM)



1. Client submits data transfer request to BeStMan instance

2. BeStMan obtains storage parameters and inquires transfer requirements

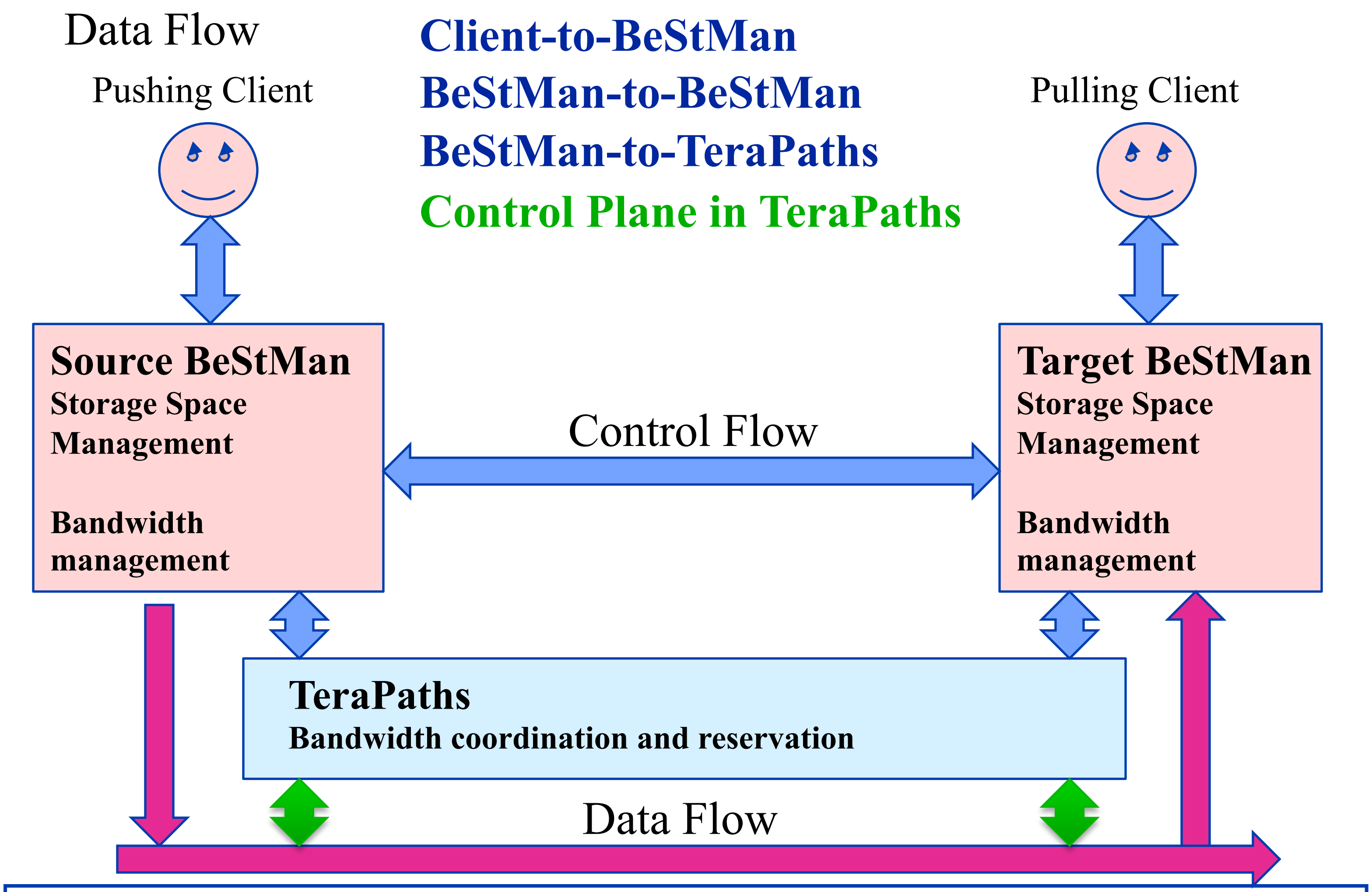
3. BeStMan sends network bandwidth reservation parameters to TeraPaths

4. TeraPaths instances negotiate end-site LAN reservation parameters

5. TeraPaths negotiates WAN reservation parameters with OSCARS

6. TeraPaths responds with negotiated reservation parameters, if solution was found, or closest suggestions

# StorNet APIs defined and functionality and Communication Flow developed



Data Flow

Pushing Client

Client-to-BeStMan

BeStMan-to-BeStMan

BeStMan-to-TeraPaths

Control Plane in TeraPaths

Pulling Client

**Source BeStMan**
**Storage Space Management**

**Bandwidth management**

Control Flow

**Target BeStMan**
**Storage Space Management**

**Bandwidth management**

**TeraPaths**
**Bandwidth coordination and reservation**

Data Flow

Notes: Push and Pull modes are needed because of security limitations

# Enhancements Needed for StorNet

- **BeStMan enhancements to:**
  - Keep track of bandwidth commitments for multiple requests
    - Both storage and network bandwidths
    - Backend database support
  - Coordinate between source and target BeStMan instances for storage space and bandwidth
  - Support advanced reservation for future time window commitments
  - Communicate and coordination with underlying TeraPaths
- **TeraPaths enhancements to:**
  - Receive network bandwidth requests from BeStMan with inputs (volume, max-bandwidth, max-completion-time)
  - Negotiate with OSCARS for "best" time window
    - "best" can be earliest completion time, or shortest transfer time
  - If success, return to BeStMan and commit reservation if BeStMan desires.
  - If failure, find closest suggestion and return to BeStMan

# StorNet API

- **Main functions:**
  - reserveRequest()
    - Input: flow specs (source/destination IPs and ports), bandwidth, start time, end time, transfer volume
    - Output: request token, reservation ids
  - commitRequest()
    - Commits the network reservation.
  - cancelRequest()
- **Auxiliary functions**
  - statusRequest()
  - extendTimeoutRequest()
    - Extends timeout if additional time is needed before committing
  - modifyRequest()
    - Modifies request parameters – primarily needed when flow specs are not known at time of reserve request

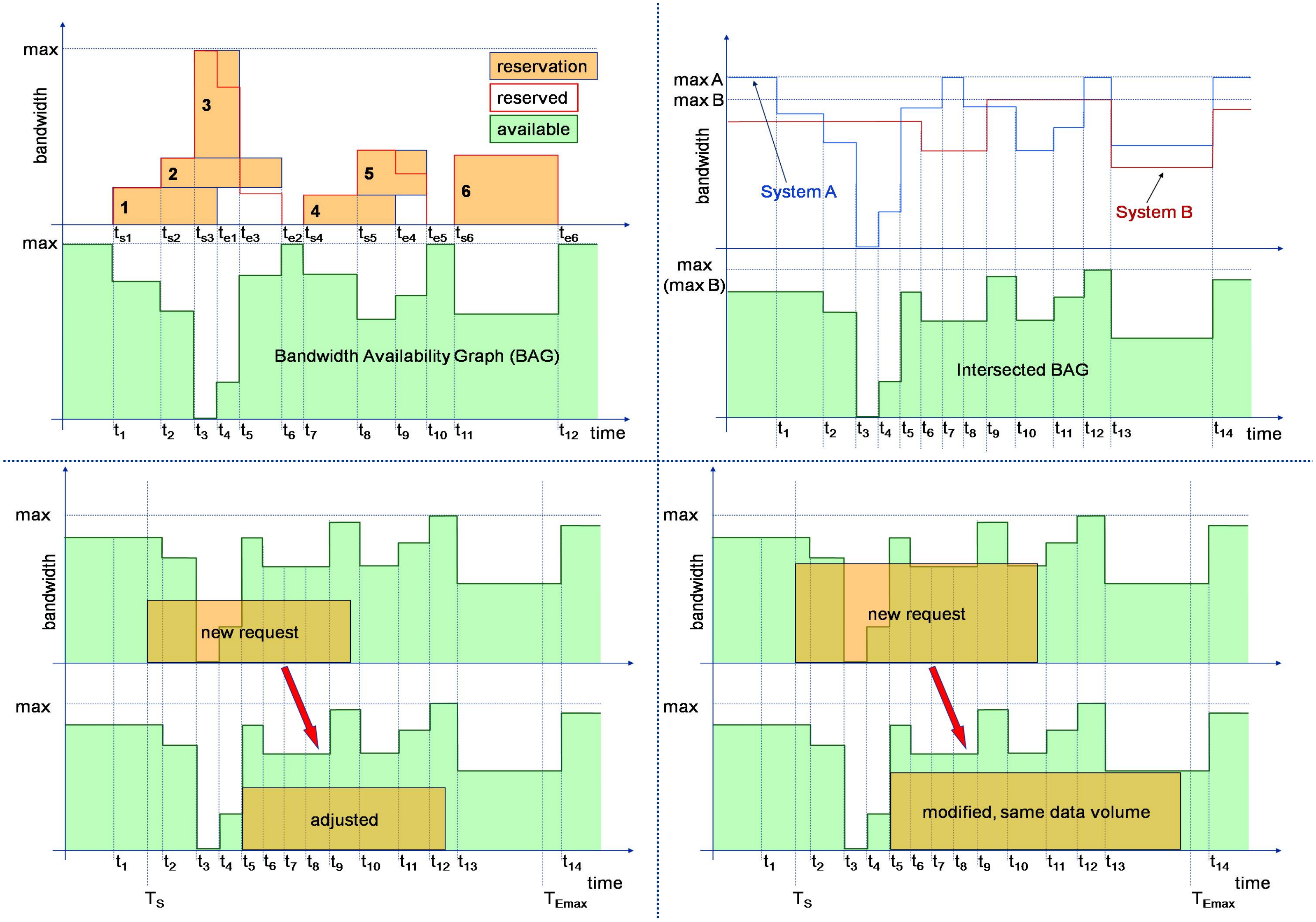
# Reservation Negotiation: Overview

- Between BeStMan and TeraPaths
  - BeStMan will send a (storage) Bandwidth Availability Graph (BAG) to TeraPaths along with the request
- Between TeraPaths end sites
  - The master TeraPaths resource manager will gather BAGs from two LANs at local end sites
  - Intersect all BAGs (from both BeStMan and TeraPaths) to form a “local” BAG
- Between TeraPaths and OSCARS
  - Obtain a ordered list of best reservations from the “local” BAG and send them to OSCARS
- Once a reservation can be made in OSCARS
  - OSCARS->TeraPaths->BeStMan

# Reservation Negotiation: Algorithms

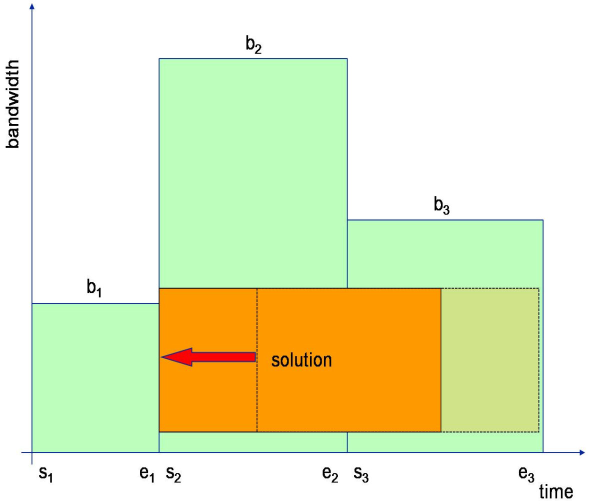
- Individual BAG can be obtained in a linear time
  - Get all reservations in [min start time,  max finish time]
  - Group them and get bandwidth usage graph (BUG)
  - Subtract BUG from the capacity and get BAG
- Intersecting BAGs can be done in a linear time
  - Similar to above

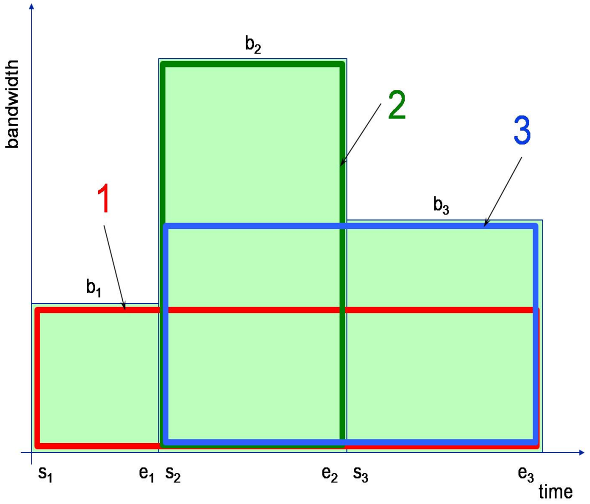
# Reservation Negotiation: Algorithms (ii)



# Reservation Negotiation: Algorithms (iii)

- Finding optimal solutions can be done in a linear time
  - Objectives : shortest duration or earliest finish time
  - Similar to problem of “The Largest Rectangle under a Histogram”
  - For each availability segment, find the largest rectangle containing it
  - Then select best solutions from those largest rectangles



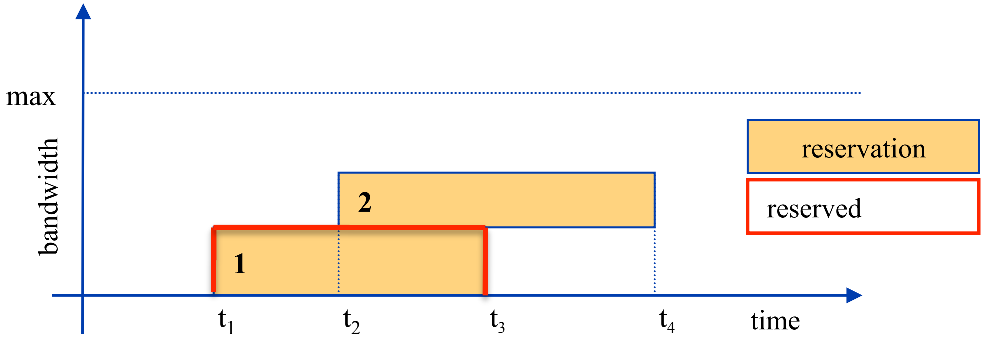


# Current Status

- Accomplished tasks
  - Design and implementation of StorNet API
  - Coordination and management of the end-to-end storage resource and bandwidth reservation
  - Management of the end-to-end network resource negotiation and configuration
  - Coordination of the storage resource service (BeStMan) with network resource provisioning service (TeraPaths)
  - Management of the negotiated end-to-end storage and network resources
  - Design and implementation of enhancements to BeStMan and TeraPaths
  - Design and implementation of intelligent multi-domain bandwidth allocation algorithms
  - Deployment and testing of basic functionality on the BNL and Univ. of Michigan testbed

# Initial test cases

- **Simple request (Single file in a request)**
  - A request that contain one file that needs to be copied from the source to the target. Storage needs to be allocated for the file, and bandwidth needs to be reserved between the sites.
  - This test was done successfully.
- **Large request (Multiple files in a request)**
  - This test case consists of many files in a request. The files need to be copied from the source to the target together.
  - This test is on going.
- **Multiple requests**
  - A multiple requests test case consists of mixture of small and large requests, so that both storage and network bandwidth reservations can overlap the requests with different capacity.



# Work in progress

- **Implementation:**
  - Additional support required for aux. calls
    - Request modifications - modifyRequest()
    - Request status query - statusRequest()
    - Request timeout extension - extendTimeoutRequest()
  - Detection/resolution of flow spec conflicts
  - Reservation negotiation
  - Support for multiple-window requests
- **Testbed:**
  - Further deployment at LBNL
- **Testing:**
  - Multiple requests with overlapping windows

# Information

- StorNet
  - http://sdm.lbl.gov/stornet/
- E-mail
  - stornet@lbl.gov