



Flexible streaming infrastructure for UNICORE

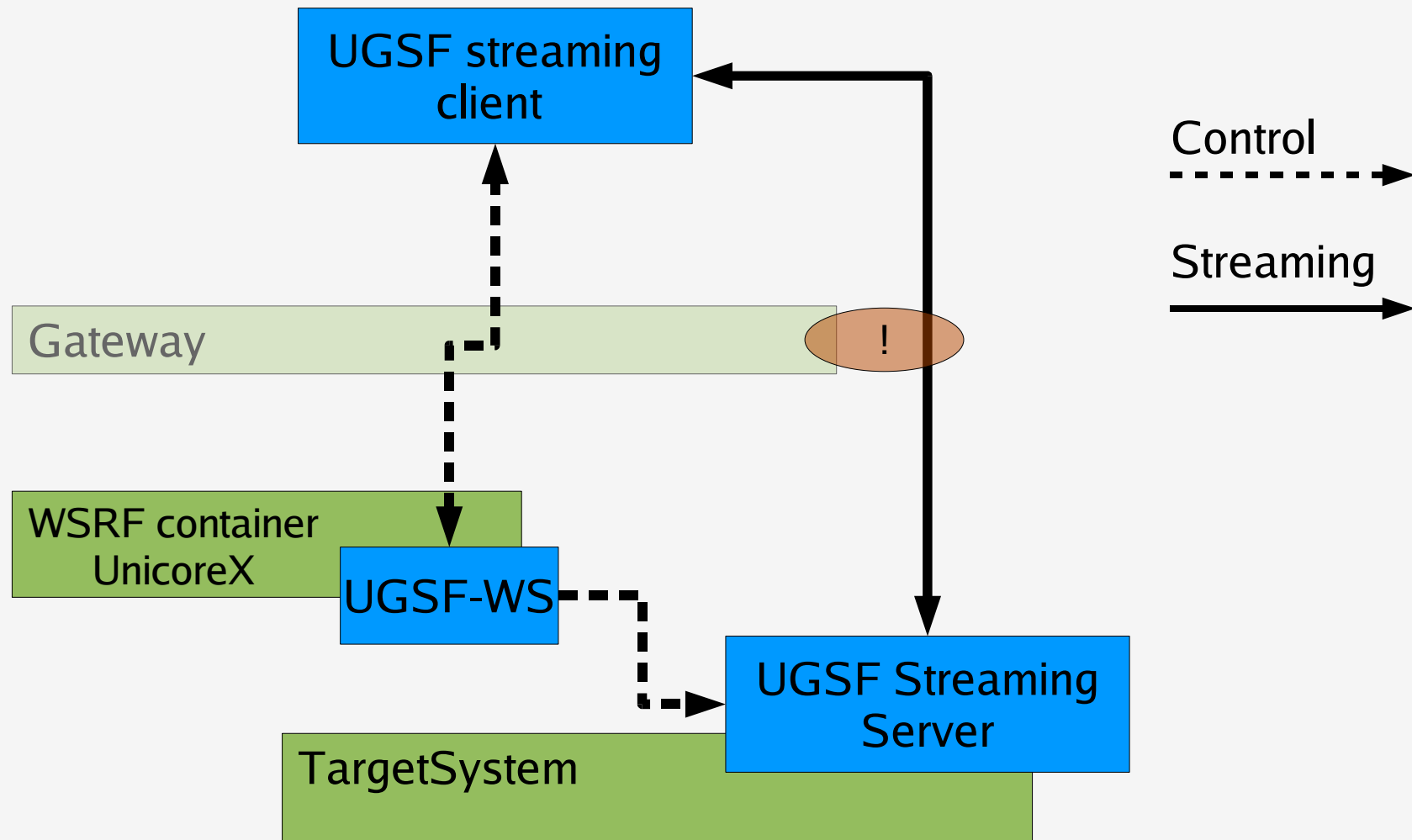
UGSF & Data Flow Client

*28 VIII 2007, K. Benedyczak, A. Nowiński, P. Bała
ICM Warsaw University*

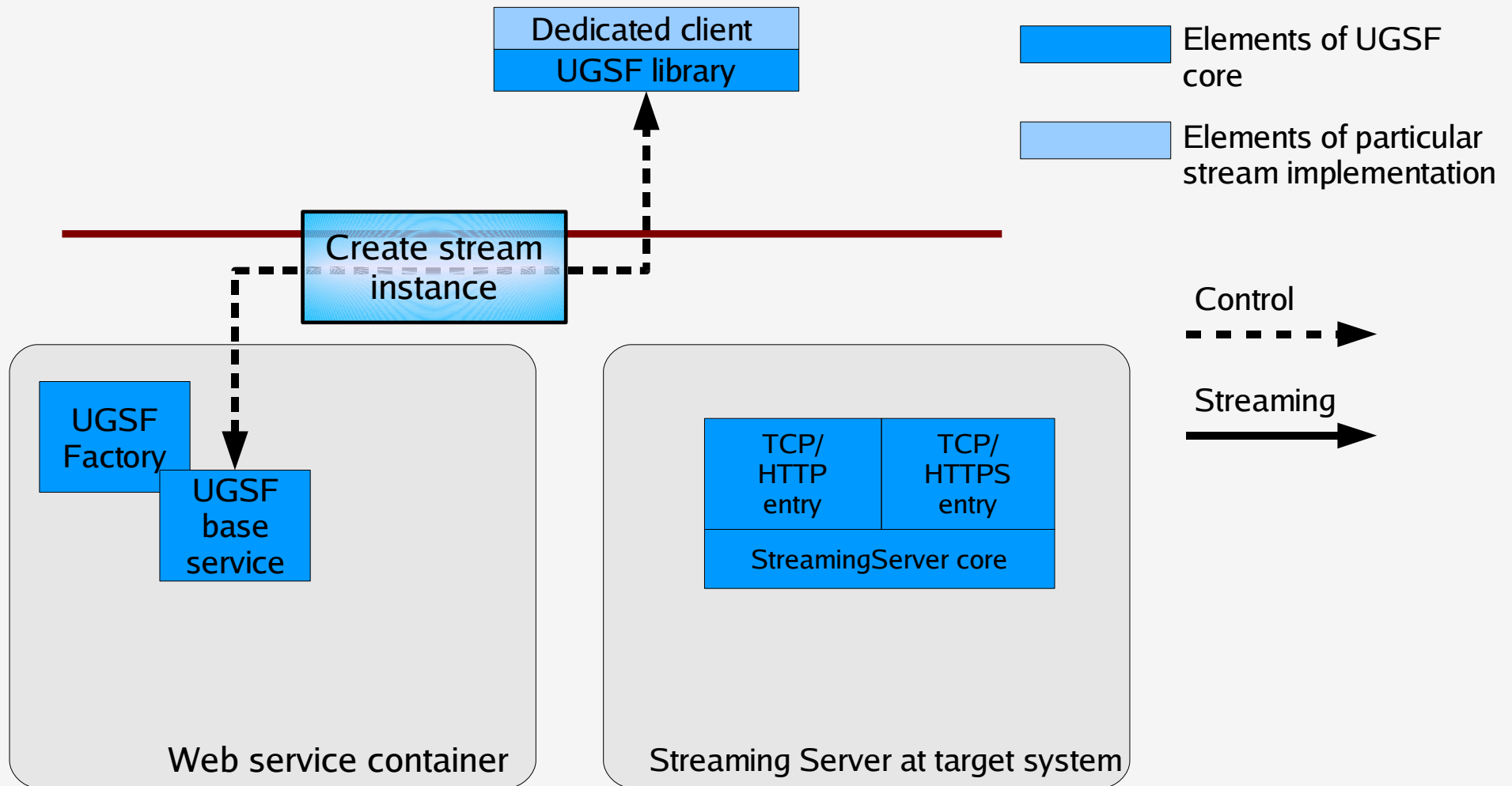
- Introduction: streaming in the Grid.
- UGSF presentation:
 - architecture, features, security and performance.
- Available stream implementations for UGSF.
- **UGSF Data Flow Client.**
- Current and future work.

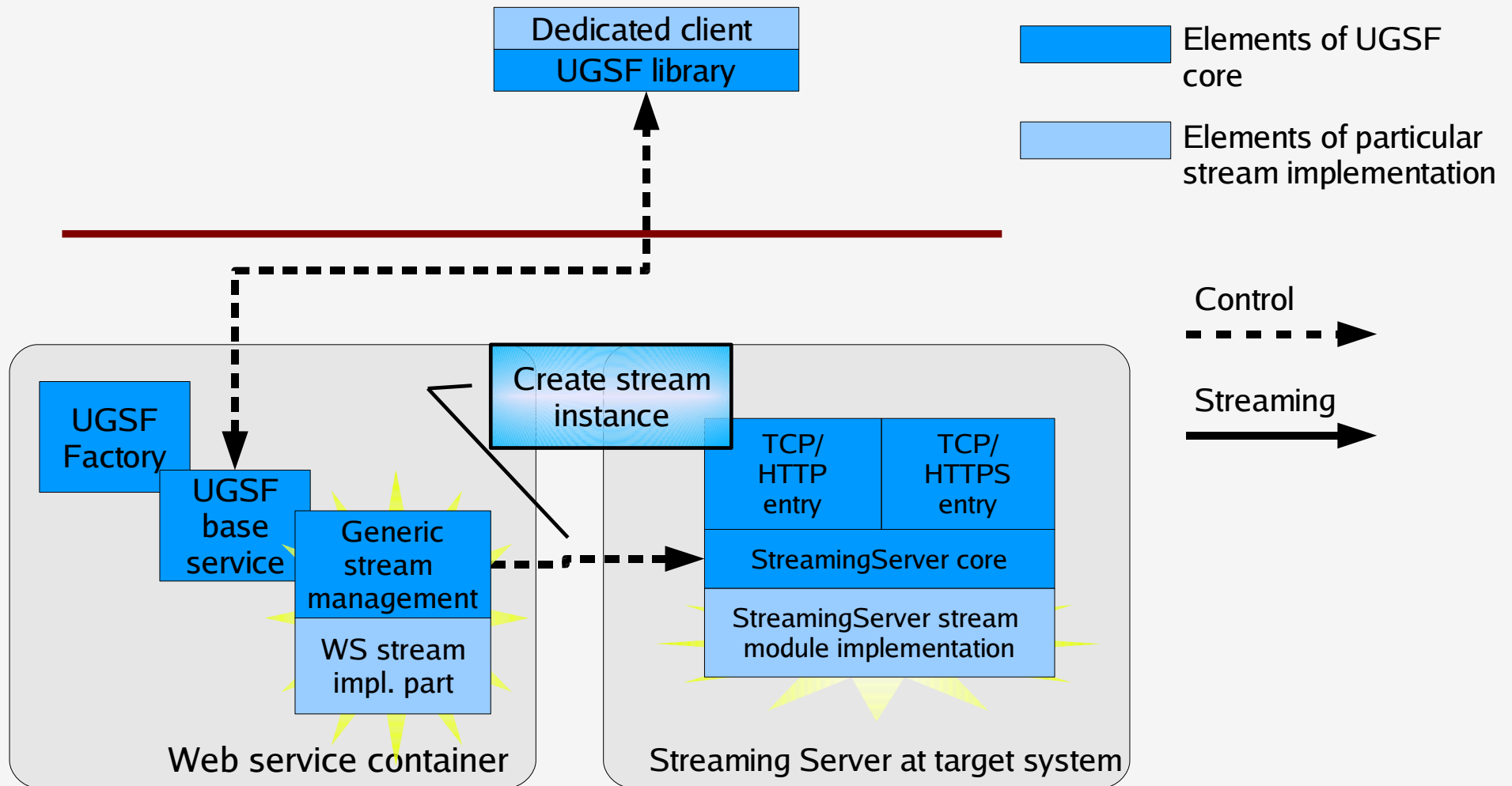
- By stream we understand data flow without *a priori* defined duration, which is processed in real time, i.e. as it is received.
- There are some efforts aiming at integration of grid and streaming concepts in more or less *universal* way:
 - NaradaBrokering and GlobalMMCS
 - E-Condor & Streamline
 - AccessGrid
 - GATES
- The most of existing solutions build separate *streaming grid* using traditional computational/data grid concepts.

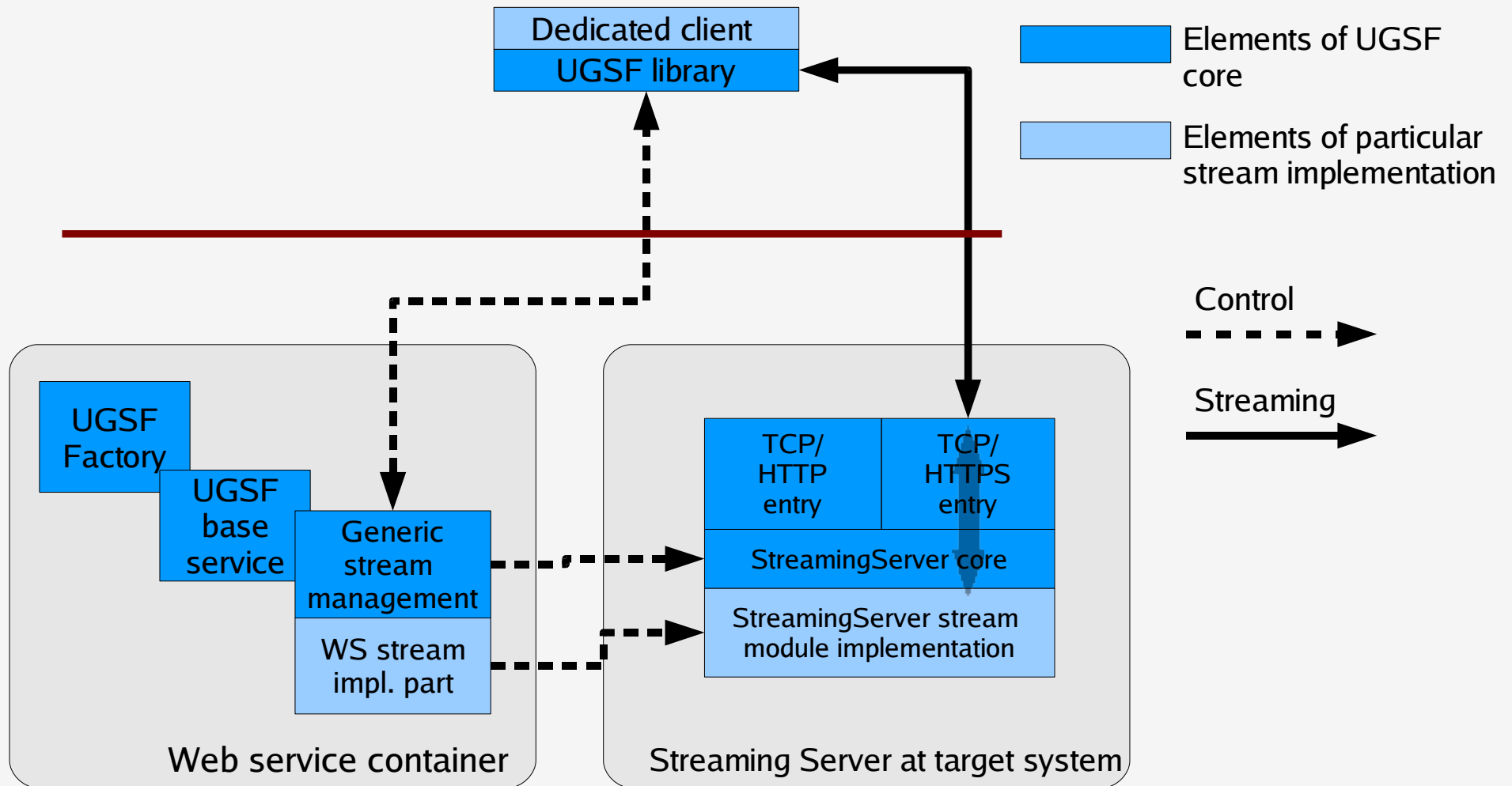
- UGSF idea is opposite to *streaming grid*:
 - To build streaming infrastructure for the *existing* computational grid middleware, in order to support streaming applications.
 - But still to be generic and universal enough to allow for creating streaming grid on top of the system if there is such need.
- UNICORE 6.0 used as platform.
- OGSA and WSRF standards employed.
- Highly extendible platform but containing ready to use components.
- Data transmission not limited to one protocol.



- **Stream Implementation:** UGSF module for streaming communication and its control.
- **Stream type:** configured and deployed by administrator stream implementation.
- **Stream instances** are created by users who want to use particular stream type. Usually they contain user's parameters. Those instances are actually existing objects that “do the job”.
- **Flow:** Every stream implementation can offer more than one “streaming connection”, which we call flow. This is handy when there are multiple inputs/outputs or we want to divide logically streamed data.









UGSF web services

- **UGSF Base Service** is factory and management point for stream instances. It also handles stream types, which are set up by administrators.
- **UGSF Stream Management Service** is used to manage stream instance. It allows for accessing all universal operations available for UGSF streams:
 - query for the stream statistics, and for the connection related informations (e.g. to discover existing data flows),
 - destroy stream, change access policy, clone flows, ...
- In practice it's extension is used which allows for accessing any of stream implementation specialized features (like 'closeFile' for FileStream).

- Is a stand-alone Java application, installed on target system, as it must have access to the streamed resources like running job's output.
- It is flexibly designed by use of modules:
 - there is module for every stream implementation,
 - there are “entry modules” which handles transport level protocol together with connection handshake.
 - Currently there are implemented TCP and TLS entry modules both using trivial HTTP for handshake.
- It is designed using Apache MINA architecture.



- UGSF Web services are secured by standard UnicoreX means. Default XACML policies are provided.
- Streaming is secured in numerous ways:
 - **Data** can be streamed over TLS. There are other possibilities as this is only dependent on entry module of UGSF streaming server, so **user** can choose desired security level.
 - **Authentication** of stream client is dependent on entry module used. E.g. in case of HTTPS X509 certificate is taken from authenticated connection.
 - **Authorization** of stream access is done based on the stream's policy, which is chosen by stream creator and can be changed later.



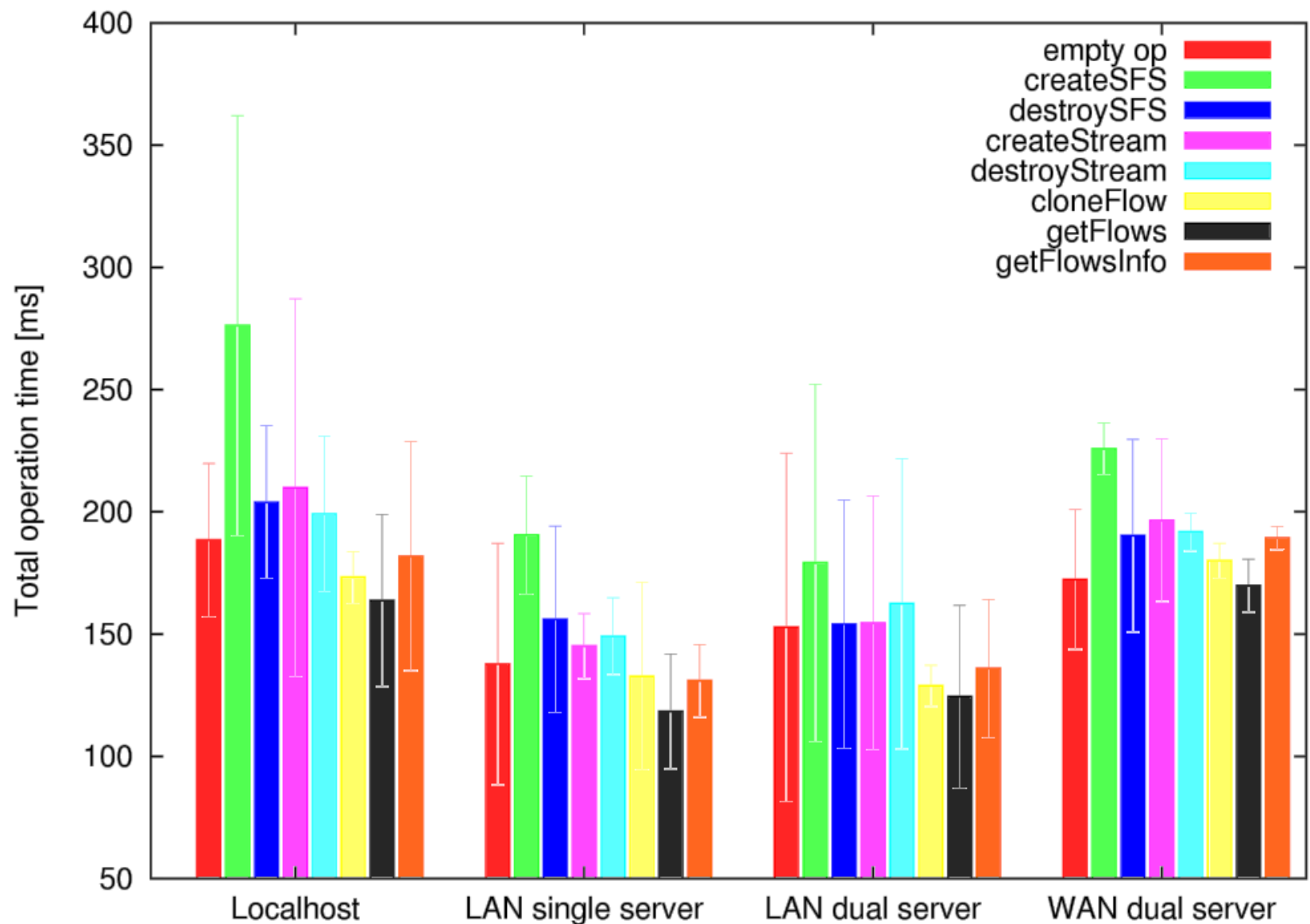
Basic streams implementations

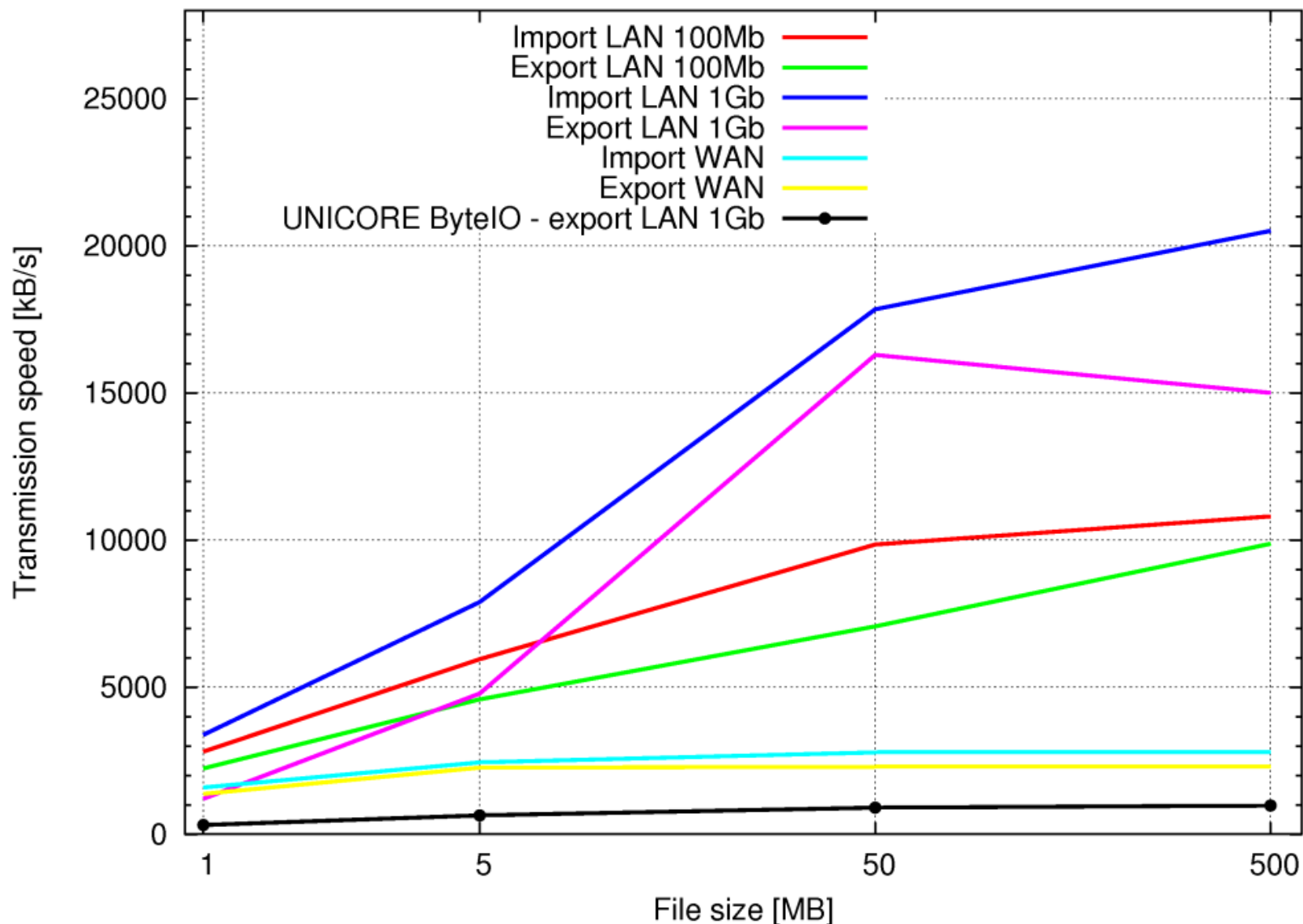
- UGSF distribution contains ready to use streams, so it can be deployed without any development.
- The attached stream implementations are of general purpose.
- ***TCPStream*** can be used to establish tunnel to TCP server working on the grid site
 - administrator defines which port is enabled in a stream type,
 - legacy services or networked devices can be simply used without any additional cost,
 - main advantage here (e.g. in comparison to SSH tunnelling) is usage of grid credentials and authorisation.



Basic streams implementations (2)

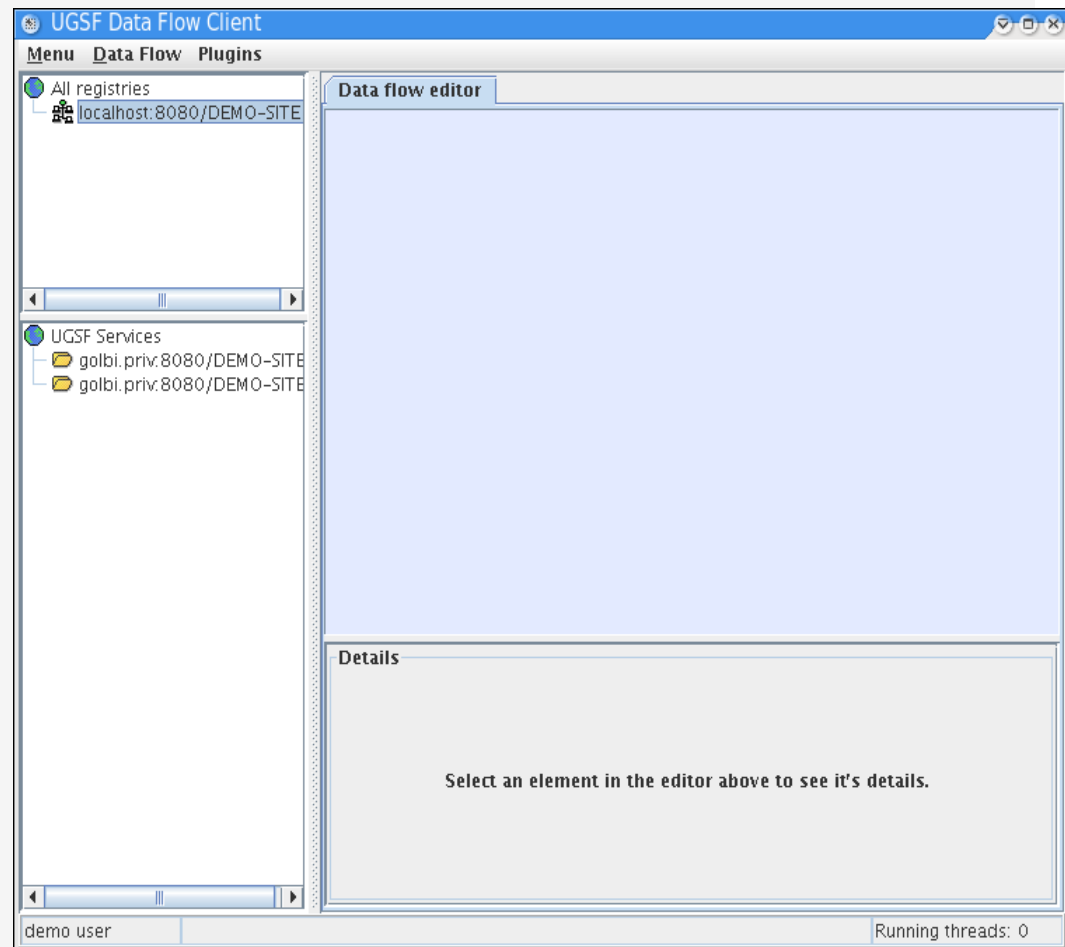
- ***FileStream*** can be used to transport files. It supports growing files: whenever file length increases additional data is streamed to a client. Supports FIFO pipes and can be used to read or write files.
- ***IVisStream*** is FileStream extension. It allows for the same as *FileStream* but for UNICORE job's files.
- ***MultiplexerStream*** duplicates it's input to many outputs. Can be used to enforce specific data flow routing.
- ***TheoraStream*** allows for decompressing OGG/Theora video streams on the fly into RAW YUV4MPEG format and vice versa.



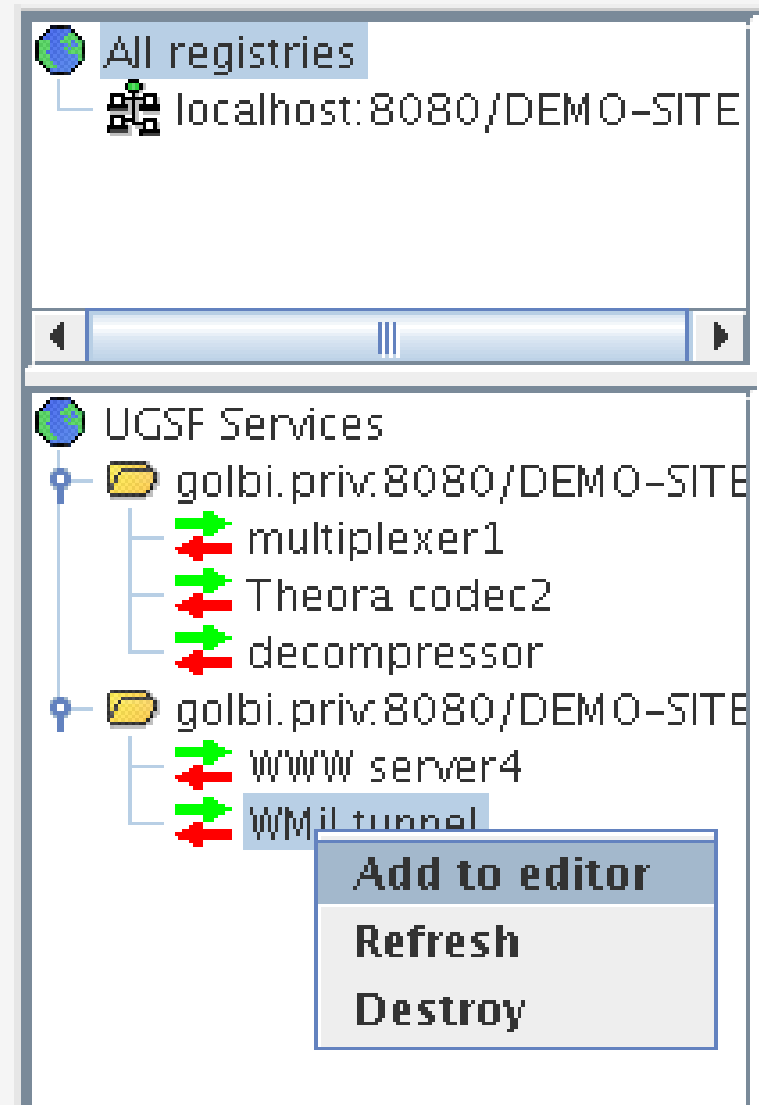


- Client library
 - based on GPE 1.4.2 for stream management,
 - access to HTTP and HTTPS entry points (in two flavours: asynchronous MINA-based and synchronous plain Java).
- Command line, interactive admin client, allowing administrators of UGSF to manage services and stream types.
- Example GPE GridBean which allows users to perform optical flow computation in streaming mode.
- Many batch utility or testing programs
- **Data Flow Client**

- Aim: GUI to allow for creating and controlling data flows consisting of arbitrary streams.
- Local machine must be able to take part in data flow.
- Targeted at power-users
 - great value for testing implementation in dozens of different conditions without development of testing programs.



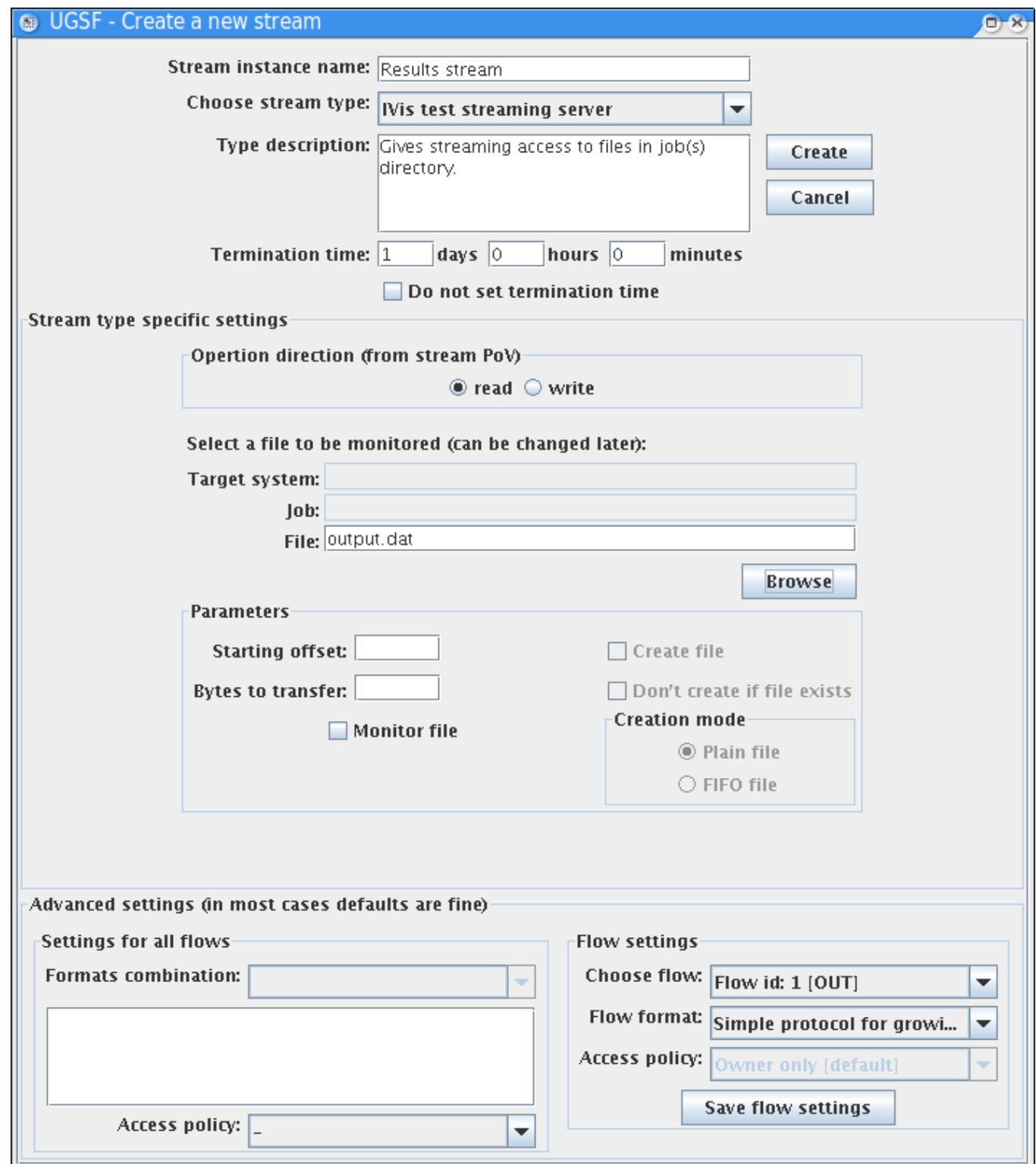
- Registries tree resembles GPE one.
- Bottom shows available UGSF services and existing (available) stream instances.
- Context menus allow for refreshing contents of registry/service and for creating or destroying stream instances.



General settings: instance name, selection of stream type and termination time.

Stream's **implementation specific parameters.** In the example IVis stream is selected, so user has to select job, it's file and optionally tune streaming settings like starting offset.

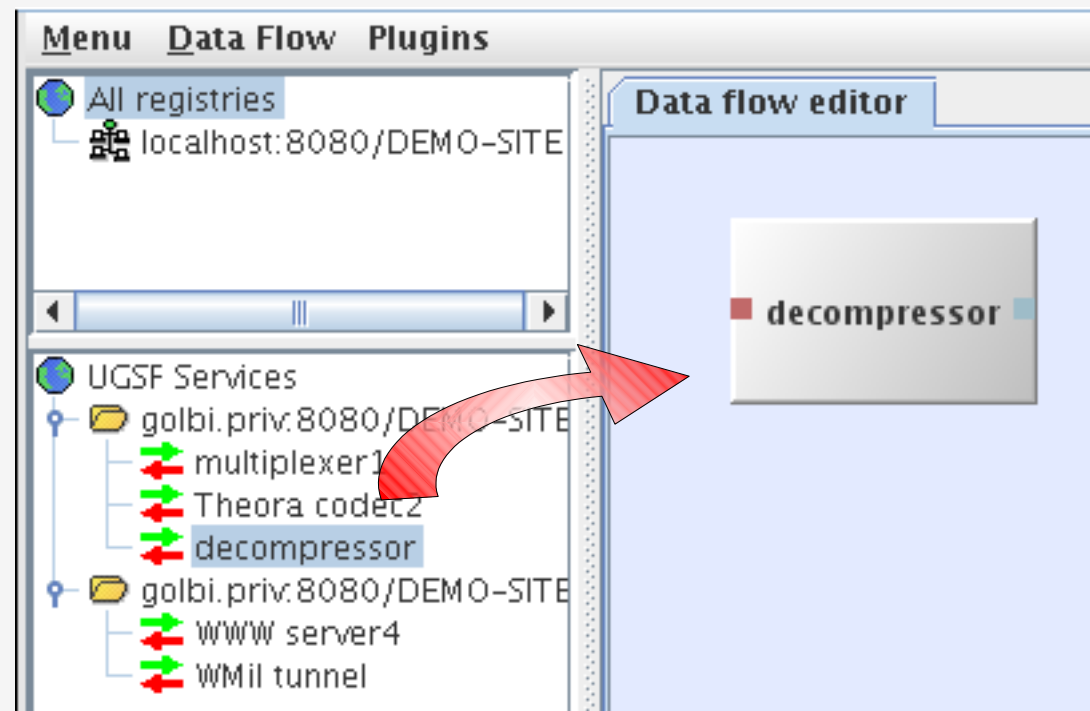
Advanced settings. All of them can be changed later. Initial formats of flows can be chosen along with access policy.

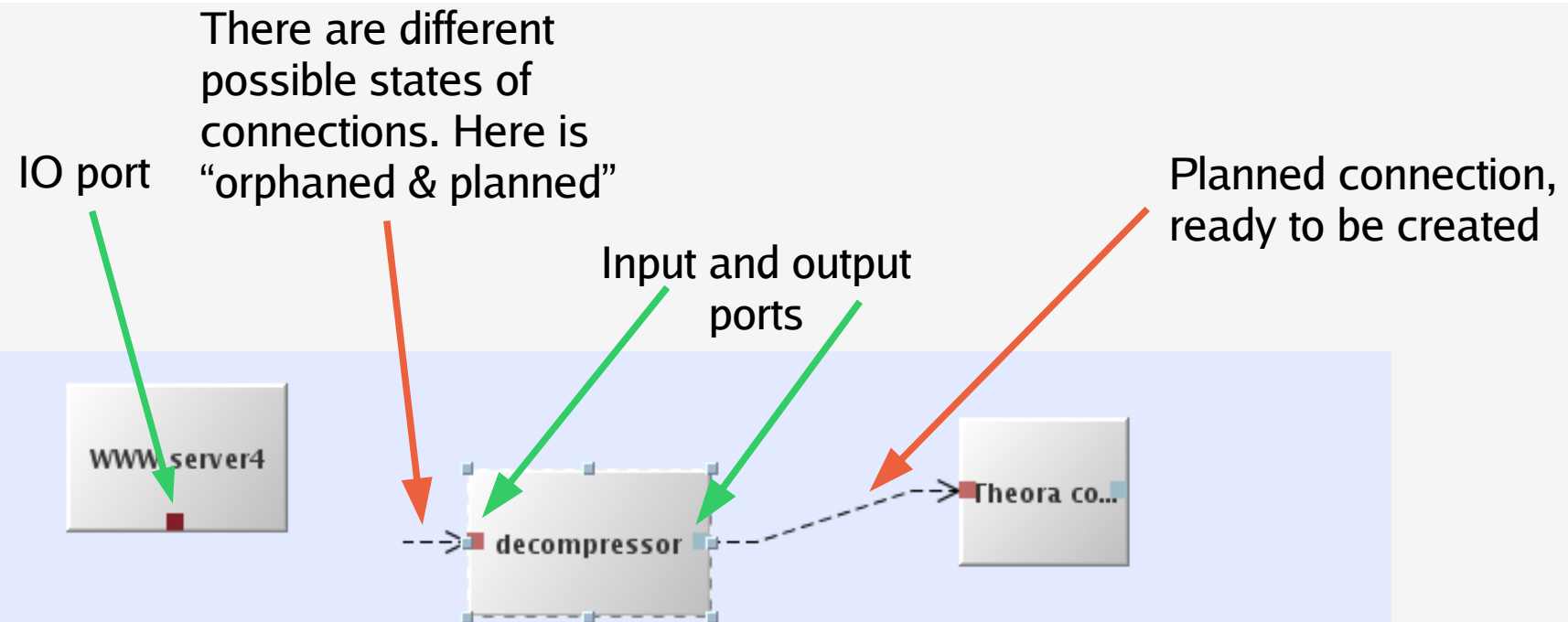


The image shows a Windows-style dialog box titled "UGSF - Create a new stream". It is divided into several sections:

- General settings:**
 - Stream instance name: Results stream
 - Choose stream type: IVis test streaming server (dropdown)
 - Type description: Gives streaming access to files in job(s) directory.
 - Termination time: 1 days 0 hours 0 minutes
 - ☐ Do not set termination time
 - Buttons: Create, Cancel
- Stream type specific settings:**
 - Operation direction (from stream PoV):
 - ☒ read
 - ☐ write
 - Select a file to be monitored (can be changed later):
 - Target system: (empty)
 - Job: (empty)
 - File: output.dat
 - Button: Browse
 - Parameters:
 - Starting offset: (empty)
 - Bytes to transfer: (empty)
 - ☐ Monitor file
 - ☐ Create file
 - ☐ Don't create if file exists
 - Creation mode:
 - ☒ Plain file
 - ☐ FIFO file
- Advanced settings (in most cases defaults are fine):**
 - Settings for all flows:
 - Formats combination: (empty dropdown)
 - Access policy: (empty dropdown)
 - Flow settings:
 - Choose flow: Flow id: 1 [OUT] (dropdown)
 - Flow format: Simple protocol for growi... (dropdown)
 - Access policy: Owner only [default] (dropdown)
 - Button: Save flow settings

- Stream instance must be added to the editor panel in order to be fully controlled.



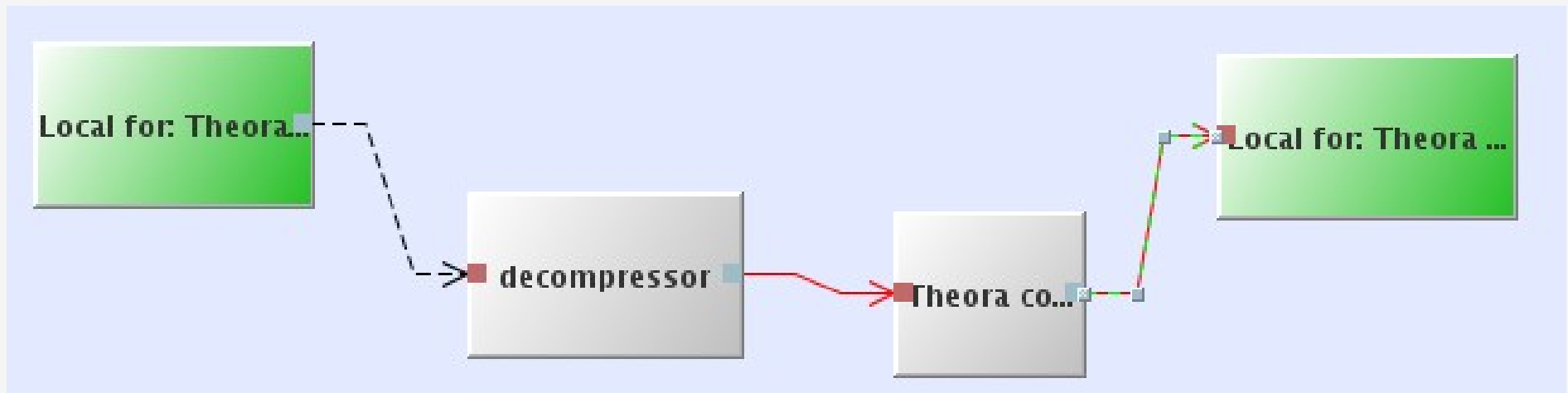


Flow details

Time of check:	8/23/07 7:44:01 PM CEST	Incoming rate:	781 B/s
Status:	CONNECTED	Received data:	118 B
Error cause:	--	Last data in:	8/23/07 7:44:01 PM CEST
Format:	YUV4MPEG uncompressed video stream [OUT]	Outcoming rate:	2529 B/s
Connection:	REMOTE_STREAM_ACTIVE	Sent data:	382 B
Remote stream:	ITE/services/TheoraStreamManagementService	Last data out:	8/23/07 7:44:01 PM CEST

Context info panel.
Here with flow info.

- Stream context menu can allow user to create local endpoint matching stream implementation in use.
- After all connections are planned (dashed lines) data flow can be instantiated. It can be done manually connection by connection, or automatically with order defined by two available algorithms (source-first and target-first).



- Whenever streams that are currently connected are added into editor there is problem with drawing “other side”.
- DFC automatically adds orphaned but existing connection edge.
- Context menu of such edge has option to discover peer stream and to add it into editor.
- There is also possibility to do this recursively in order to discover the whole data flow at once.



- DFC plugins allows for using stream specific features.
- Plugins can provide:
 - panel used to enter parameters used during stream instance creation,
 - context menu items for the whole stream and every of it's flows,
 - implementation of local endpoint, which can stream to/from remote stream instance.
- Some simple stream implementations like MultiplexerStream do not need any plugin as the whole functionality is modelled in UGSF core primitives.

- Development of higher level infrastructure on top of UGSF which will support users and DFC to manage streaming data flows: finding appropriate stream types automatically, saving/restoring data-flows, ...
- Even better integration with grid job submission services(?)
- Performance tuning of MINA based communication.
- Development of other entry points: e.g. parallel TCP, high performance messaging paradigms (WS-Eventing/WS-Notification).

This work was supported by European Commission under IST grant UniGrids (No. 004279).

UGSF Data Flow Client

MenuData FlowPlugins

All registries

localhost

UGSF Service

golbi.priv

multi

Theor


decor

golbi.priv

www

WMil

UGSF video stream



The END

thank you

for attention

Pause

Local for: ...

decompressor

Theora co...

Local for: ...

Time of check:8/23/07 7:46:10 PM CEST

Status:DISCONNECTED

Error cause:--

Format:YUV4MPEG uncompressed video stream [IN]

Connection:NOT_CONNECTED

Remote stream:--

Incoming rate:5568 kB/s

Received data:377 MB

Last data in:8/23/07 7:44:01 PM CEST

Outcoming rate:1 B/s

Sent data:118 B

Last data out:8/23/07 7:45:10 PM CEST

Refresh

☐ Monitor

delay [s]:

5

15

25

35

45

55

demo user

Running threads: 0