Proposed Scenarios for NSI-WG

Petr Holub, <hopet@ics.muni.cz>

March 2009

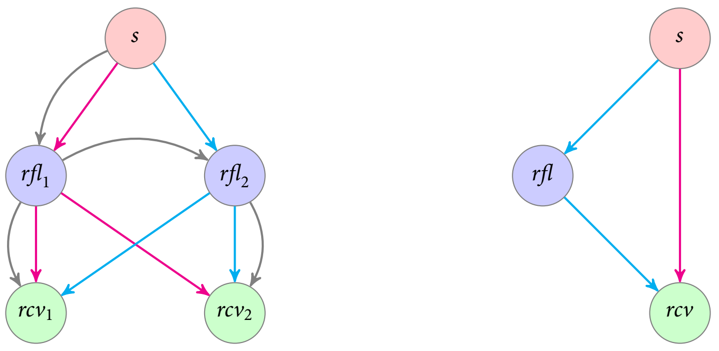
The scenarios proposed in this document are based on our experiences with

- *UltraGrid* – low latency HD and post-HD video transmission system, with supported data bit rates ranging from 250 Mbps to 1.5 Gbps (the range will expand in the future) and end-to-end latency (with 2 m of fiber) at 80–170 ms. `http://ultragrid.sitola.cz`
- *CoUniverse* – self-organizing environment for application and network orchestration, primarily developed to support real-time collaboration. `http://couniverse.sitola.cz`
- *VirtCloud* – virtual network allocation for virtual computing clusters. `http://meta.cesnet.cz`

The CoUniverse examples given below are based on case when CoUniverse orchestrates media streaming applications like CoUniverse, where multi-point data distribution is implemented using reflectors (instead of relying on multi-point data distribution capabilities of the underlying network).

1 Resource Co-Allocation

For some applications (e.g., CoUniverse orchestrating reflectors), some network applications can be allocated on different network nodes and their roles are interchangeable. In a simple scenario shown in Figure 1a, there are one sender s that can send to one destination only, two nodes rfl_1 and rfl_2 that can act as multi-point distribution reflectors, and two receivers rcv_1 and rcv_2.



(a) (b)

Figure 1: Network and resource co-allocation and an example of alternative scenarios.

There are four possible ways to allocate the nodes and links to fulfill the requirement to distribute data from the s to the two receivers rcv_1 and rcv_2, with three of them being directly shown in Figure 1a:

- (magenta) rfl_1 is used distribute the data (e.g., when rfl_2 is unavailable or one of the following links is unavailable: $s \rightarrow rfl_2$, $rfl_2 \rightarrow rcv_1$, $rfl_2 \rightarrow rcv_2$),
- (cyan) rfl_2 is used distribute the data (e.g., when rfl_1 is unavailable or one of the following links is unavailable: $s \rightarrow rfl_1$, $rfl_1 \rightarrow rcv_1$, $rfl_1 \rightarrow rcv_2$),
- (gray) both rfl_1 and rfl_2 are used distribute the data, e.g., when $rfl_1 \rightarrow rcv_2$ is unavailable. There is another scenario symmetric to this one with rfl_1 and rfl_2 functionality swapped.

Even in the case of simple setup shown in Figure 1b, when the link $s \rightarrow rcv$ is unavailable, the traffic can be still delivered if $s \rightarrow rfl$, $rfl \rightarrow rcv$ links and reflector on rfl node are co-allocated.

From network perspective, the network nodes and network links need to be co-allocated, based on availability of both the nodes and links. While CoUniverse framework could do the co-allocations from the application level, attempting to allocate all possible data distribution scenarios combinatorially results in too many allocation requests. Further the first feasible solution may not be optimum from network scheduling perspective. It would be also possible to try to find the optimum network topology on the application level, if the network provides its topology description with information on what network links and capacities are already allocated. There are however inherent risks in this approach, like different applications doing scheduling in parallel leading to incompatible allocation requests and impossibility for the applications to see up-to-date map of the network as the allocated capacities/links always keep changing.

2 Multi-Point Connections

Some applications need to build a multi-point network, where nodes can communicate with any other node within the created network. In case of VirtCloud, we're creating virtual networks for clusters built of virtual machines running user-provided images. The applications need to be able communicate within the virtual cluster in arbitrary way as the goal is to provide user with virtualized infrastructure which simulates a fully interconnected private cluster. Building a network supporting this behavior using point-to-point circuits would basically require allocating $n^2 - n$ links, where n is number of nodes.

CoUniverse can also benefit from multi-point connections being provided by the network. Because of knowledge of the orchestrated applications, that CoUniverse has, hints can be provided about planned data flows in the network compared to the general VirtCloud example above. Multi-point connections can be combined with alternative paths (and related co-allocation problem) shown in Figure 1b, thus allowing for more flexible network allocation.

After the multi-point network is allocated, the both VirtCloud and CoUniverse would benefit from learning the allocated network structure: VirtCloud can propagate this information to the user so that he can take this into account when running his applications, while CoUniverse can utilize this directly to optimize data flows.

3 Notes on Specifying Link Parameters

It is important that the rigorous link specification parameters are explained to the users and there are tools to help find out problems when they occur. An example I have given during NSI WG meeting in Catania is requested bandwidth. The user may be able to specify average bandwidth but doesn't have tools to find out maximum burst size produced by the application. If the application produces any bursts larger than the average bandwidth, the excessive packets are lost when if the link capacity specification is strict and there is not sufficient buffering to smooth out the burst. While application developers (may) have understanding of these effects and they may have good reasons to create bursts[1], the users may run into serious problems when requesting bandwidth for such applications. The higher bandwidth used, the more pronounced the problem becomes as larger buffers may be needed to smooth out the bursts.

The danger is in the fact that user of such bandwidth-constrained service may not be willing to use it and pay for the service: when doing monitoring of the interface on sender and receiver with commonly used tools (measuring time averages over, e.g., 1 s periods), it sees requested bandwidth going out from the sender and only part of the traffic being received on the sender—thus making the network is broken from the user perspective. This is obviously wrong from the network perspective, but monitoring tools, capable of detecting (clipped) bursts, need to be provided to help the user with identification what network capacity needs to be applied for.

Another risk in case of the bursty traffic with strict capacity allocations is that the user may not be willing to pay for the service that has enough headroom to accommodate the bursts: from the user perspective, the capacity is highly over-provisioned, poorly utilized and the remaining capacity could have been used for some other applications.

[1]E.g., end-to-end latency minimization in case of UltraGrid, because the video frame grabber data source is discrete source of data that gives whole video frames to the application at 24 Hz, 25 Hz, or 30 Hz frequency (depending on source frame rate) and then the frame needs to be transported to the receiver as fast as possible. For practical reasons to cope with lower-performance 10GE cards, burst limited to 6 Gbps in UltraGrid based on busy waiting (CPU inefficient). Average bandwidth is 1.5 Gbps in such case.