CoUniverse: Framework for Building Self-Organizing Collaborative Environments Using Extreme-Bandwidth Media Applications

Miloš Liška
Faculty of Informatics
Masaryk University
Botanická 68a, 621 00, Brno
Czech Republic
xliska@fi.muni.cz

Petr Holub
Faculty of Informatics
Masaryk University
Botanická 68a, 621 00, Brno
Czech Republic
hopet@ics.muni.cz

Abstract

Using internet for media-based real-time collaboration has become widespread activity. In this paper, we present a framework called CoUniverse, designed for building real-time user-empowered collaborative environments to work primarily on high-speed networks with true high-bandwidth applications such as uncompressed high-definition video. The system is designed for unreliable experimental infrastructures and therefore its operation relies heavily on self-organizing principles—this is also useful approach for extending it to larger infrastructures. When media stream bitrate is comparable to capacity of the links, the additive assumption no longer holds and the system needs to have sophisticated scheduling. Concept of scheduler is a flexible plug-in for the CoUniverse framework and in this paper, we present a formal scheduling model based on constraint programming including evaluation of its prototype implementation. CoUniverse is designed to utilize external media application, so that wide variety of existing tools can be used. The whole system has been prototyped and demonstrated, e. g., during international demonstration on GLIF 2007 workshop.

1 Introduction

Collaboration among peer-humans always involved some element of self-organization [1]. The same principle has also been followed by virtual collaborative environments to some extent. Majority of the systems is however fairly static in a sense of reacting to different events occurring both inside and outside of the system. E.g., media tools, that are relying on multicast, are depending on its availability. H.323 tools are usually able to react to changing available network bandwidth by adjusting media compression parameters—but if the central interconnecting MCU breaks, there is usually not much to do except for terminating the call.

Recent development of systems like AccessGrid or VRVS EVO (see Related work in Section 6) introduces another level of autonomous behavior. Users can either influence some system parameters manually (e.g., choosing between unicast and multicast in case of the AccessGrid) which is followed by some automatic reconfiguration, or the system attempts to react automatically to some events, e. g., by choosing another reflector for data distribution among participants of a videoconference.

This approach is however insufficient for high-end collaborative applications, where the media streams are comparable with the capacity of the link and thus where implicit additive assumption ("sending one more stream on a link does no harm to the whole system") is no longer valid even on backbone network links. The whole system requires more complex scheduling in order to achieve reasonable performance or even to work at all. In order to interact with advanced networking features like lambda-services [2], much richer interfaces are also needed.

Another aspect of the self-organizing collaborative environments is how much power should be given to its users. In our previous work (e. g., [3]), we have argued in favor of user-empowered approach to the largest extent that is possible. Proposed self-organizing system should also follow this paradigm and utilize various schemes of organization that don't require any administrative privileges over nodes or networks. However, this approach can be extended further in a way that allows users to modify behavior of the system—but it also raises the question what can be exposed to the user in order to maintain the cooperative nature of the whole system? And what should be the interfaces, that would be acceptable and actually useful for the users?

In this paper, we propose a framework for such collaborative environments called *CoUniverse*. Basic design principles used for proposing architecture of CoUniverse are discussed in Section 2. Resulting proposed architecture including overview of basic components and organization of the network is described in Section 3. The system has been prototyped including preliminary version of the scheduler as discussed in Section 4. The system has already been demonstrated during several events and its evaluation especially with focus on performance of current version of the scheduler is in Section 5. Because the field of collaborative environments is rapidly moving forward, we brief related work in Section 6. The paper is concluded by tackling future research tasks in Section 7.

2 Design Principles

CoUniverse design focuses on the following major areas:

1. self-organization of collaborative environment,
2. incorporation/encapsulation of external tools (even those which don't support the middleware directly),
3. continuous adaptation on changing conditions based on built-in monitoring of applications, nodes, and network links,
4. support for media streams with bitrate comparable to capacity of links (including advanced scheduling of data streams to links),
5. visualization of the environment for the users to make it understandable.

The CoUniverse is organized as one or more *collaborative Universes*, where the actual collaboration takes place, and a *Multiverse*, used for registration and lookup of clients and Universes. The collaborative Universes are intended to accommodate collaborative groups of limited sizes[1] and thus can implement functionality that may be hard or impossible to deploy at large. This includes features like sophisticated scheduling and aggressive monitoring of components and network that provides basis for fast reaction to problems that may occur. On the contrary the Multiverse provides very limited functionality, it has to scale well with respect to large number of participating nodes.

In terms of self-organization, the system should be capable of reacting to events in the system, namely to events raised by users, nodes, and by the monitoring. It includes applications being started/terminated, network links being turned up/down, changes in link parameters (capacity, loss, latency, jitter), nodes being added to and removed from the Universe and nodes being reconfigured.

Because the CoUniverse is designed to integrate high-bandwidth applications, it is necessary to interface with services provided by advanced networks like lambda services [2] or network resource allocators [4]. The CoUniverse includes notion of a lambda link: it is a link that may be down because it is not allocated and may be allocated prior to being used, provided there is an identity participating inside the collaborative Universe, that has sufficient privileges to do so.

CoUniverse needs to have a scheduler to support applications with media streams comparable to network link capacity. The scheduling objectives may vary: for simple interactive applications with fixed quality, it usually includes minimization of media distribution latency and possibly minimization of number of nodes involved in the network. For more complex applications where quality is an adjustable parameter, maximization of the quality may also be included. Output of the scheduler has to include not only the plan itself, but also a *workflow* describing how to implement the plan, as there are many functional dependencies. For instance, network links need to be allocated prior to starting media applications that will send data over them.

The whole system should follow the user-empowered paradigm as much as possible: there should be no need to have administrative privileges especially over the network and components. The system should be able to run entirely in user space.

Programmability by Users In later stages, we would like to study programmability and "debuggability" of the environment—this is a topic which manifests in all self-organizing systems that are not just very simple or single purpose with one size fits all solution. This is different from single purpose self-organizing environments like Skype or file sharing networks. The limited size of Universe also enables assumption that withing each Universe, users behave intentionally in a cooperative manner and thus we ignore problems with intentional cheating, especially when the environment becomes programmable. Even within this scenario, impact of programmability needs to be studied with respect to isolation of individual Universes. An example of programmability, that is already present in CoUniverse framework, is user-replaceable scheduler. User-defined scheduler may take into account more or less information than the default one, depending on intents of the community working inside the specific Universe. Another example, that is rather very simple from this perspective as it has close to zero impact on self-organization capabilities of the network, is programmability of virtual floor as shown for Isabel in [5].

[1] Further information on importance, formation and dynamics of small groups can be found in [1].

3 Proposed Architecture

3.1 Network Organization

As discussed above, CoUniverse is organized as one or more collaborative Universes and a Multiverse. From the networking point of view each Universe consists of a *control plane* used for control communication of all components of the Universe and one or more *data planes* used for actual data exchange between Universe components. Both control plane and data planes are forming an overlay networks on top of actual physical network infrastructure.

Multiverse and control planes of collaborative Universes are based on P2P networking substrate which provides necessary robustness for the Multiverse and control planes. Moreover a P2P substrate provides functions like clients and Universes description, naming and addressing, lookups and reliable data transfers.

Data planes of the collaborative Universes are based on available physical networking infrastructure. The data planes are optimized for maximum performance and minimum latency when transmitting data between the components of the Universe. As data planes are virtual overlays over physical networking substrate, they exist only in case when there is an Application Group (see bellow) to utilize it. The system is designed with user-empowered paradigm in mind and thus it naturally relies on using application-level media "routers" and distributors (reflectors, Active Elements [3]) for multipoint data distribution instead of being dependent on network-native multicast[2].

3.2 Collaborative Universe

Collaborative Universes, as shown in Figure 1, consists of nodes, each of which runs Universe Peer client. Universe peers are providing a base for communication among the Universe components, managing underlying node configuration and steering media applications configured on the very node. Nodes within the Universe are aggregated into *network sites*, usually representing all nodes of a single site participating in the collaborative Universe. To give more precise definition, a network site is a set of collocated nodes, where each site may have one or more users participating. Expressed using terminology defined below, typical property of all nodes within one site is that there are no consumers consuming data from producers from the same site (this definition doesn't include media distributors).

Each network node is configured by specifying *(i)* a list of its physical network interfaces and their parameters, *(ii)* a list of Media Applications which are installed on the node, and *(iii)* a network site the node belongs to. Media Application is any application which is used to create the collaborative environment and which produces or consumes a media stream (e. g., videoconferencing clients, audioconferencing clients, data distributing Active Elements (AE) [7], etc.). All Media Application producers (except AEs) are producing exactly one media stream which is then sent to exactly one consumer.

Media applications are organized into Application Groups (AG). AGs are then generalizing a particular functionality of the collaborative environment (e. g., audio or video conferencing, desktop sharing etc.). Media applications within an AG are orchestrated using Application Group Controller (AGC). AGC is a service running on top of at least one of the regular Universe peers. The purpose of AGC is to collect node configurations from all peers within the collaborative universe, assemble a topology of universe data planes, invoke a scheduler to schedule media streams of media applications to physical network links, create a configuration for each media application based on scheduled media streams and finally send the configuration together with data plane topology to respective universe peer. The universe peer in charge then adjusts the configuration of steered media application so that it corresponds to respective scheduled media stream.

Scheduler within the AGC is invoked either manually (especially for the first time) or automatically as a reaction to a change Collaborative Universe state (e. g., new node appeared, a node is not reachable using a particular network link etc.).

3.3 Monitoring

Each Universe peer comprises monitoring of steered media applications, network links of physical networking substrate which might be used to build a data plane for media applications and network links that are actually part of some data plane. Monitoring of the data planes network links is more aggressive than monitoring of network links of generally available physical networking substrate since the links of data planes are actually used for media applications data exchange. At the same time, the links that are not used in any of the Universe data planes need to be monitored less frequently just so that the AGC eventually has a notion of their state when some event in the Universe occurs and

[2]Further discussion on why network-native multicast is *not* a user-empowered solution and why its virtualization, e. g., using overlay networks, is needed, can be found in our previous work [3]. With multicast as a layer-3 network service, users are dependent on network administrators—who usually consider multicast as second-class citizen compared to unicast routing. This is also evidenced by shift to "reflectors" in many collaborative communities: AccessGrid (`http://www.accessgrid.org/`), ResearchChannel iHDTV (`http://ihdtv.sourceforge.net/`), or Microsoft ConferenceXP (`http://www.conferencexp.net/`). There are also other good reasons to implement overlay networks, e. g., for user-empowered NAT traversal for collaborative environments [6] or per-user processing.

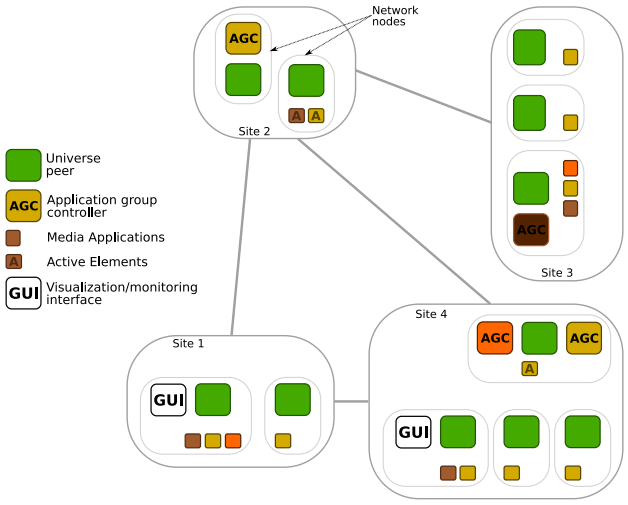

Figure 1. Scheme of the Collaborative Universe with its components. Different colors for Media Application squares mean different media applications.

those links might be used for some newly scheduled media streams.

3.4 Visualization

Visualisation gives an overview of actual collaborative Universe state to the user. Our goal is to provide a dynamic visualisation displaying on one hand topology of the physical network between nodes of the collaborative Universe, which might be used to build the data planes, and on the other hand active (currently scheduled) media streams. Visualisation of active media streams is extremely useful especially when incorporating data from network and applications monitoring. Moreover users can also easily find out whether the schedule chosen for a given network topology has the desired effect (i.e., users can see, talk to, or collaborate with each other in the way it was intended in a particular collaborative universe).

3.5 Scheduling Network Model

In order to describe the scheduling algorithms implemented into the CoUniverse framework, we need to introduce formal notation first. In this section, only the notation is described, while the actual constraints used for scheduling are available in Section 4.

Let I be a set of all network interfaces, $i \in I$ a network interface. Furthermore let N be a set of all nodes in the Universe, $n \in N$ a particular node. Then $\mathrm{node}(i) = n$ where $n \in N$ is a node n with configured network interface i.

Let $l = (i, j)$ be a network link for $i, j \in I$. Then $L = I \times I$ denotes a set of all network links and we denote a particular network link as $l \in L$. We can define following properties of a network link l: $\mathrm{begin}(l) = i$ such that $l = (i, j) \wedge i, j \in I$ is the originating interface i of the link l, $\mathrm{end}(l) = j$ such that $l = (i, j) \wedge i, j \in I$ is the ending interface j of the link l. $\mathrm{cap}(l)$ denotes link capacity.

Finally, let P be a set of producers where $p \in P$ is a media application producer, C a set of consumers where $c \in C$ is a media application consumer and M set of media distributors where $m \in M$ is an Active Element (AE). Producers, consumers and media distributors are running on nodes $n \in N$. Let $\mathrm{consumers}(p)$ where $p \in P$ be a set of consumers for a particular producer p. Thus $\bigcup_p \mathrm{consumers}(p)$ is a set of all active consumers, i.e., those that have requested a data stream from some producer. In the opposite direction, $\mathrm{producer}(c)$ is the requested producer for consumer c. Furthermore we define $\mathrm{node}(p) = n$ where $n \in N \wedge p \in P$, $\mathrm{node}(c) = n$ where $n \in N \wedge c \in C$ and $\mathrm{node}(m) = n$ where $n \in N \wedge m \in M$ as a parent nodes of producer p, consumer c and media distributor m. Media application producer $p \in P$ is producing a media stream with minimal bandwidth $\mathrm{min\_b}(p)$ and maximal bandwidth $\mathrm{max\_b}(p)$.

4 Prototype Implementation

Prototype implementation of CoUniverse uses a current stable version of JXTA [8] P2P framework to implement CoUniverse control plane. Both Multiverse and collaborative Universes are implemented as user name and password authenticated private JXTA peer groups separated from public JXTA P2P network. Current implementation of Multiverse lacks most of the functionality mentioned in previous section and is used just for Universe registration and static lookup.

Current prototype implementation of CoUniverse uses just one AGC to orchestrate all applications within the collaborative universe. We are using a single AGC to simplify the implementation and to avoid synchronization issues between several AGCs running at the same time. We implemented an interface for steering of generic media applications. In the current implementation of CoUniverse, the Universe Peer is able to control a variety of UltraGrid flavors [9] for both uncompressed and compressed full 1080i HD video transmissions—compared to description in [9], bitrates from 250 Mbps to 1.5 Gbps are now also supported, based on several compression and bitrate reduction algorithms. Amongst other supported applications are: VideoLan Client[3] for HDV video transmissions, VIC[4] for low

[3] http://www.videolan.org/

bandwidth videoconferencing (used as a fallback for building of the collaborative environment) and RAT[4] tool for audio transmissions.

Media streams scheduler was implemented as constraint-based solver using a Choco solver library[5]. The solver searches for a solution which is a mapping of media streams on network links. Formally we are looking for set of *stream links* $SL = L \times P$. Scheduler plans the stream links so that $(l,p) = 1$ where $(l,p) \in SL$ for a stream link that is planned to be actively used for data distribution in the Universe and $(l,p) = 0$ where $(l,p) \in SL$ for unused stream link. For sake of brevity in the text below, we say that stream link (l,p) *exists* iff $(l,p) = 1$.

Speaking in terms of network model given in previous section the constraints for the solver look as follows:

- Stream links constraints
  - $\forall (l,p) \in SL$ such that $\mathrm{node}(\mathrm{begin}(l)) = \mathrm{node}(p)$ holds $\mathrm{min\_b}(p) < \mathrm{cap}(l)$
    Parent network link l of the stream link must have sufficient capacity to transmit the media stream p.
  - $\forall (l,p) \in SL.\ \exists p \in P$ such that $\mathrm{node}(p) = \mathrm{node}(\mathrm{begin}(l)) \vee \exists m \in M$ such that $\mathrm{node}(m) = \mathrm{node}(\mathrm{begin}(l))$
    Each stream link must have producer or media distributor on its beginning node.
  - $\forall (l,p) \in SL.\ \exists c \in \mathrm{consumers}(p)$ such that $\mathrm{node}(c) = \mathrm{node}(\mathrm{end}(l)) \vee \exists m \in M$ such that $\mathrm{node}(m) = \mathrm{node}(\mathrm{end}(l))$
    Each stream link must have a consumer receiving data using the stream link.
- Producer constraints
  - $\forall p \in P.\ \left\| \{c \in C \mid c \in \mathrm{consumers}(p)\} \right\| > 1 \Rightarrow \neg\exists (l,p) \in SL$ such that $\mathrm{node}(p) = \mathrm{node}(\mathrm{begin}(l)) \wedge \mathrm{node}(c) = \mathrm{node}(\mathrm{end}(l))$
    More than one consumer for a particular producer means that there cannot be any direct stream link between consumers and respective producer as the producer has to send the media stream through at least one media distributor.
- Consumer constraints
  - $\forall c \in \bigcup_p \mathrm{consumers}(p)$ exists just one $(l,p) \in SL$ such that $\mathrm{node}(c) = \mathrm{node}(\mathrm{end}(l))$
    Media stream for each active consumer is received using exactly one stream link.
  - $\forall c \notin \bigcup_p \mathrm{consumers}(p)$ exists no $(l,p) \in SL$ such that $\mathrm{node}(c) = \mathrm{node}(\mathrm{end}(l))$
    There are no media streams for any of inactive consumers (i.e., those that hasn't requested any data from any producer).
  - $\forall c \in \bigcup_p \mathrm{consumers}(p).\ \exists p \in P$ such that $\exists (l,p) \in SL$ where $\mathrm{node}(p) = \mathrm{node}(\mathrm{begin}(l)) \wedge \mathrm{node}(c) = \mathrm{node}(\mathrm{end}(l))$ and $p = \mathrm{producer}(c)$.
    or
    $\forall c \in \bigcup_p \mathrm{consumers}(p).\ \exists m \in M$ such that $\exists (l,p) \in SL$ where $\mathrm{node}(m) = \mathrm{node}(\mathrm{begin}(l)) \wedge \mathrm{node}(c) = \mathrm{node}(\mathrm{end}(l))$
    Each active consumers has to be covered by the requested producer either directly or through some media distributor.
- Data distribution tree constraints
  - $\forall p \in P.\ \left\| \{c \in C \mid c \in \mathrm{consumers}(p)\} \right\| = 1 \Rightarrow \left\| \{(l,p) \mid (l,p) \in SL \wedge (l,p) = 1\} \right\| \in [1, \left\|M\right\| + 1]$
    Number of used stream links for producers with only one consumer is greater or equal to number of producers. That means data may go either directly, or through some forwarding media distributor (typically in case that direct sending is not available for one reason or another). Number of stream links obviously must not exceed number of all the media distributors in the network plus one.
  - $\forall p \in P.\ \left\| \{c \in C \mid c \in \mathrm{consumers}(p)\} \right\| > 1 \Rightarrow \left\| \{(l,p) \mid (l,p) \in SL \wedge (l,p) = 1\} \right\| \in [\left\|\mathrm{consumers}(p)\right\| + 1, \left\|M\right\| + \left\|\mathrm{consumers}(p)\right\| + 1]$
    Minimal number of used stream links is greater or equal to number of consumers for give producer plus one for multipoint data distribution. Upper bound number of consumers for given producer plus number of all the media distributors plus one.
- AE constraints
  - $\forall m \in M$ exists just one $p \in P$ for all $(l,p) \in SL$ such that $\mathrm{node}(m) = \mathrm{node}(\mathrm{begin}(l)) \vee \mathrm{node}(m) = \mathrm{node}(\mathrm{end}(l))$
    A single media distributor instance can only serve for distribution of data from a single producer.
  - $\forall m \in M.\ m$ distributes data from $p \implies \neg\exists c \in \mathrm{consumers}(p)$ such that $\mathrm{node}(m) = \mathrm{node}(c)$
    Media distributor is not scheduled together with another consumer for the same producer on a single node.
  - $\forall m \in M$ holds $\left\| \{(l,p) \in SL \mid \mathrm{node}(m) = \mathrm{node}(\mathrm{begin}(l))\} \right\| \geq \left\| \{(l,p) \in SL \mid \mathrm{node}(m) = \mathrm{node}(\mathrm{end}(l))\} \right\|$

[4] http://mediatools.cs.ucl.ac.uk/nets/mmedia/
[5] http://choco-solver.net/

There has to be at least the same number of egress media streams as ingress media streams for particular AE.

- Link capacity constraint
  - $\forall l \in L, \forall (l,p) \in SL$ holds $\sum_{p} \text{max\_b}(p) \cdot (l,p) < \text{cap}(l)$ where $(l,p) = \{0,1\}$
    That is bandwidth requirements of all the scheduled stream links (l,p) must not exceed capacity of the link l the stream links are bound to.

Another available constraint is based on elimination of intra-site links (i.e., links (l,p), where $\text{node}(\text{begin}(l))$ and $\text{node}(\text{end}(l))$ belong to the same site). This can speed up scheduling up to $10\times$ for many scenarios, but it may also disable useful solution, e. g., those where media distributors are collocated in the same site with producers and/or consumers. In case of need, this can be however circumvented by moving media distributors to a separate site.

Based on its settings, the solver can return just a single first match solution or a solution optimized for a minimal media streams distribution latency between the nodes. Based on the network topology and configured media applications the solver may also return a number of equivalent (and even optimal) solutions. In such case the first solution is used and deployed within the collaborative universe.

Because Java lacks any reliable tools for network connectivity monitoring, we have implemented a custom client-server based ping tool. The tool measures not only availability of peers through the native network, but also network round-trip time, which is an important parameter for latency minimization in our scheduler model. Each universe peer is running the server part implicitly and then is pinging all other known universe peers. In section 3, we mentioned that we need more aggressive monitoring of those network links which are part of some data plane and are used for media applications data exchange than of those network links which are just generally available in the network substrate. This is implemented by a priority and default classes of links which are monitored. A ping client is invoked each second for each network link with scheduled media stream (which is put into the priority class) and each 10 seconds for network links in the default class.

In our prototype, we implemented a semi-static visualization (see Figure 2) of the collaborative Universe. The visualization is updated with every new scheduling of media streams within the Universe. Currently the visualization shows only active scheduled media streams with some rudimentary description and static parameters of the media streams. However, even such a simple visualization is helpful to check that the collaborative Universe is started up and configured as was intended to.

Figure 2. Visualization of scheduled stream links in CoUniverse GUI during GLIF 2007 demonstration.

Java sources and JAR archive of CoUniverse prototype implementation are downloadable from `https://www.sitola.cz/CoUniverse`.

5 Prototype Implementation Evaluation

Performance and scalability of the CoUniverse environment heavily relies on the scheduler, therefore we have performed a number of simulations with various network topologies and data distribution schemes and measured the time necessary to obtain a schedule for given network topology and distribution scheme.

We chose a full mesh m:n, 1:n tree and direct 1:1 data distribution schemes as a test cases for evaluation of scheduler performance and scalability. Network topologies were given by the data distribution schemes and number of sites in the collaborative universe. The m:n distribution scheme test case topology was generated so that each site had one node with UltraGrid media application producer a node with UltraGrid consumer for each other site and a node with AE. This scenario simulates full-mesh collaboration among peers. The 1:n tree distribution test case was generated so that one site had a UltraGrid producer node and UltraGrid consumers node for all other sites in the topology, every other site comprised of one UltraGrid producer node and one UltraGrid consumer node. This scenario is realistic, e.g., for virtual classroom type environment, where the lecturer gives his talk in multiple remote rooms in parallel. A corresponding number of AE nodes was generated with respect to the fact that one AE can replicate 1,5 Gbps media stream from an UltraGrid producer to at most 6 UltraGrid

consumers where 10 Gbps network link is available. Finally direct 1:1 data distribution was a simple test case with generated pairs of UltraGrid producer nodes and UltraGrid consumer nodes, where each UltraGrid consumer was receiving the media stream from a particular preconfigured UltraGrid producer. This is sort of artificial scenario to show scalability limits. All nodes had one 1 Gbps and one 10 Gbps network interface configured in all three test cases.

All measurement results were obtained on a 2 GHz Pentium M machine with 1 GB of RAM running a Linux operating system. A 1.2.05 version of Choco solver library was used. Table 1 shows excerpt of measured times necessary to find feasible plans for above mentioned test cases with Choco solver set up to return all feasible solutions while Table 2 shows corresponding times measured for Choco solver set up to return just the first feasible solution and exit immediately. The results are summed up in graph 3 and 4 showing that Choco solver scales reasonably for 1:n and direct 1:1 data distribution schemes with up to 25 nodes in the network topology. The worst scheduler performance was observed for m:n data distribution scheme. For such a scheme we were able to obtain a schedule in a reasonable amount of time for up to 12 nodes aggregated into 3 sites.

Table 1. MatchMaker evaluation (all feasible solutions)

Distribution scheme	Sites	Nodes	Network links	Media applications	Active Elements	Scheduling time [s]
m:n	2	6	60	6	2	0,178
m:n	3	12	264	12	3	0,510
m:n	4	20	760	20	4	1970,540
1:n	2	5	40	5	1	0,181
1:n	4	11	220	11	1	0,360
1:n	6	17	554	17	1	0,753
1:n	7	20	760	20	1	1,110
1:n	8	24	1104	24	2	8,914
1:n	10	30	1740	30	2	37,299
1:n	12	36	2520	36	2	105,308
1:1	2	4	24	4	0	0,187
1:1	5	10	180	10	0	0,333
1:1	8	16	480	16	0	0,979
1:1	11	22	924	22	0	2,009
1:1	14	28	1512	28	0	3,344
1:1	17	34	2244	34	0	6,160
1:1	20	40	3120	40	0	10,161
1:1	23	46	4140	46	0	21,661

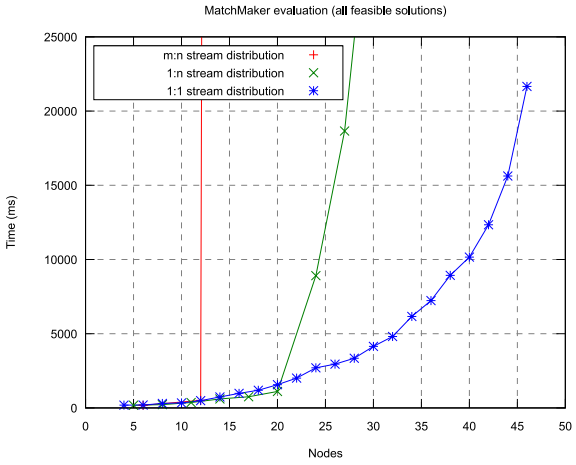

Figure 3. Time to find all feasible solutions for given network topology.

Table 2. MatchMaker evaluation (first feasible solution only)

Distribution scheme	Sites	Nodes	Network links	Media applications	Active Elements	Scheduling time [s]
m:n	2	6	60	6	2	0,308
m:n	3	12	264	12	3	0,447
m:n	4	20	760	20	4	1986,047
1:n	2	5	40	5	1	0,169
1:n	4	11	220	11	1	0,285
1:n	6	17	554	17	1	0,758
1:n	7	20	760	20	1	0,924
1:n	8	24	1104	24	2	3,747
1:n	10	30	1740	30	2	17,518
1:n	12	36	2520	36	2	69,907
1:1	2	4	24	4	0	0,187
1:1	5	10	180	10	0	0,343
1:1	8	16	480	16	0	0,862
1:1	11	22	924	22	0	1,900
1:1	14	28	1512	28	0	3,382
1:1	17	34	2244	34	0	5,745
1:1	20	40	3120	40	0	9,727
1:1	23	46	4140	46	0	21,637

Another important metrics is the time necessary to establish the collaborative environment. Therefore we measured the time relative to Universe Peer startup needed to connect

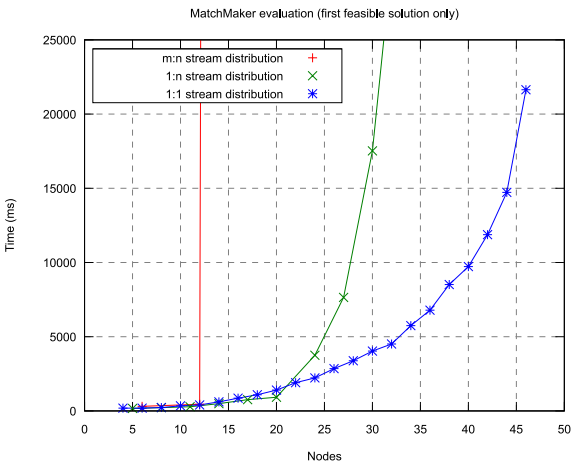

Figure 4. Time necessary to find only first feasible solution for a given network topology.

to a Universe. We observed three different cases where in the first one Universe Peer was connecting to AGC running on a local machine, seconds a Universe Peer was connecting to an AGC running in a local network and finally we measured a time necessary to connect to AGC[6] running on a node in our laboratory network for a Universe Peer running on a network node at Louisiana State University. Measured times necessary for Universe Peer to connect to AGC and to get a first response from AGC are showed in Table 3. Longer connection times in later two cases were caused by longer time needed to lookup the AGC in the Universe using the JXTA control plane. All measurements were obtained using version 2.4.1 of JXTA protocol implementation.

Table 3. Connection times and response time for Universe Peer and AGC.

	local node AGC	*local net AGC*	*remote net AGC*
connect to AGC	26,6 s	30,2 s	30,3 s
response from AGC	27,4 s	31,2 s	34,4 s

5.1 Demonstrations

Prototype implementation of CoUniverse was evaluated during a demonstration at GLIF 2007 meeting in Prague in September 2007. CoUniverse was used to orchestrate a network of twelve nodes using high quality, high bandwidth HD video transmissions and audioconferencing to create multipoint-to-multipoint collaborative environment connecting three sites (Louisiana State University, USA with Charles University, Czech Republic and Academia Sinica, Taiwan).

Creating such a collaborative environment means in praxis configuring and steering of more than two dozens of media applications and Active Elements to bring up media streams connecting all sites. Configuring all media applications and AEs at dozen of machines presents huge amount of manual work which is overwhelming for users of such environment. Moreover there must be at least one user of the collaborative environment having precise idea how to create the media streams between all media applications and AEs based on knowledge of available physical network substrate between all participating sites. Last but not least the users are not able to ensure resiliency and fast recovery of such environment in case of any network, node or media application failure, because it might mean even newly configuring of all nodes and applications.

Both issues were well addressed deploying CoUniverse. Although creating node configurations for all nodes in the Universe is initially quite time consuming as well, users have to create just local configurations describing network interfaces of the local machine and location of local media applications.

The system was also demonstrated during SuperComputing'07 event in Reno, NV in November 2007.

6 Related Work

As mentioned in the introduction, some extent of self-organization is usually built into the all but the simplest collaborative tools. H.323 and SIP tools that are considered a sort of industrial standard as a videoconferencing platform can accommodate changes in available link capacity by changing compression parameters of media streams. Isabel [10] platform has similar properties by means of flow server and also features programmable floor control [5], which is however on the level of GUI programmability only.

Probably closest to CoUniverse idea is currently VRVS EVO [11], which allows self-organization of the collaboration network. It is however a closed system that doesn't incorporate external tools and namely it can't cope with media streams that have bandwidth requirements comparable with link capacity. From user perspective, VRVS EVO can be viewed as a system similar to Skype.

Another important videoconferencing platform is AccessGrid [12], which doesn't have any self-organizing properties. The fail-over mechanisms are only very simple and have to be initiated by the user, e. g., by selecting unicast media transport instead of multicast. Compared to the other

[6] In order to connect to AGC Universe peer must connect to given JXTA super-peer, authenticate into Universe peer group, obtain AGC advertisement containing AGC identification within the JXTA network and send local node configuration to AGC.

systems, it may seem simple, but it follows several of CoUniverse design principles which the other systems are not compliant with: user-empowered paradigm at least for the collaborative system components (which are open-source and may be installed by end-users arbitrarily) and it is also extensible and incorporates external applications.

7 Conclusions and Future Work

In this paper, we have designed a framework for advanced self-organizing collaborative environments called CoUniverse. The system is targeted to incorporating high-end multimedia tools while utilizing advanced high-speed networks with their specialized services.

Even though we have already implemented and successfully demonstrated a prototype of the CoUniverse, it still leaves many unanswered questions stated in the introduction to this paper. One big issue is optimization of the scheduling algorithms in order to support larger infrastructures. It should also better utilize knowledge of network structure, even if it is only partial. We want to include scheduling for native multipoint applications. Another issue that needs to be further investigated is programmability of the whole system by its users. This is also important in the context of the scheduler, which may need to be able to incorporate user-defined constraints on its behavior. From practical perspective, it needs to be integrated with a some more sophisticated monitoring framework in order to obtain more precise information from the environment, namely from the underlying networking layer (e. g., better capacity and network topology estimates).

Another topic which is of interest is providing some role-based security scheme for the CoUniverse, as current prototype has only very limited security mechanisms incorporated.

Acknowledgments

This project has been kindly supported by the research intent "Parallel and Distributed Systems" (MŠM 0021622419).

We are also thankful to our colleague David Antoš for his help with formalizing the network model used in our work and LaTeX typesetting of the math used in this paper.

References

[1] Holly Arrow, Joseph E. McGrath, and Jennifer L. Berdahl. *Small Groups as Complex Systems*. Sage Publications, 2000.

[2] Franco Travostino, Joe Mambretti, and Gigi Karmous-Edwards. *Grid Networks: Enabling Grids with Advanced Communication Technology*. John Wiley & Sons, 2006.

[3] Petr Holub, Eva Hladká, and Luděk Matyska. Scalability and robustness of virtual multicast for synchronous multimedia distribution. In *Networking - ICN 2005: 4th International Conference on Networking, Reunion Island, France, April 17-21, 2005, Proceedings, Part II*, volume 3421/2005 of *Lecture Notes in Computer Science*, pages 876–883, La Réunion, France, April 2005. Springer-Verlag Heidelberg.

[4] Jon MacLaren. Co-allocation of compute and network resources using harc. In *Proceedings of Lighting the Blue Touchpaper for UK e-Science: closing conference of ESLEA Project*, volume PoS(ESLEA)016, 2007.

[5] Juan Quemada, Tomás de Miguel, Santiago Pavon, Gabriel Huecas, Tomas Robles, Joaquín Salvachúa, Diego Andres Acosta Ortiz, Vicente Sirvent, , and Fernando Escribano. Isabel: An application for real time collaboration with a flexible floor control. In *CollaborateCom 2005*, 2005.

[6] Petr Holub, Eva Hladká, Michal Procházka, and Miloš Liška. Secure and pervasive collaborative platform for medical applications. *Studies in Health Technology and Informatics*, 126:229–238, 2007.

[7] Eva Hladká, Petr Holub, and Jiří Denemark. An active network architecture: Distributed computer or transport medium. In *3rd International Conference on Networking (ICN'04)*, pages 338–343, Gosier, Guadeloupe, March 2004.

[8] Bernard Traversat, Ahkil Arora, Mohamed Abdelaziz, Mike Duigou, Carl Haywood, Jean-Christophe Hugly, Eric Pouyoul, and Bill Yeager. Project JXTA 2.0 super-peer virtual network. `http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf`.

[9] Petr Holub, Luděk Matyska, Miloš Liška, Lukáš Hejtmánek, Jiří Denemark, Tomáš Rebok, Andrei Hutanu, Ravi Paruchuri, Jan Radil, and Eva Hladká. High-definition multimedia for multiparty low-latency interactive communication. *Future Generation Computer Systems*, 22(8):856–861, 2006.

[10] T. P. De Miguel, S. Pavon, J. Salvachua, and J. Quemada Vives. ISABEL—experimental distributed cooperative work application over broadband networks. *Lecture Notes in Computer Science*, 868:353–362, 1994.

[11] Philippe Galvez. Evo: Enabling virtual organizations. In *CHEP'07*, Victoria, Canada, 2007.

[12] L. Childers, T. Disz, M. Hereld, R. Hudson, I. Judson, R. Olson, M. E. Papka, J. Paris, and R. Stevens. ActiveSpaces on the Grid: The construction of advanced visualization and interaction environments. In Björn Engquist, editor, *Simulation and visualization on the grid: Parallelldatorcentrum, Kungl. Tekniska Högskolan, seventh annual conference, Stockholm, Sweden, December 1999: proceedings*, volume 13 of *Lecture Notes in Computational Science and Engineering*, pages 64–80, New York, NY, USA, 2000. Springer-Verlag Inc.