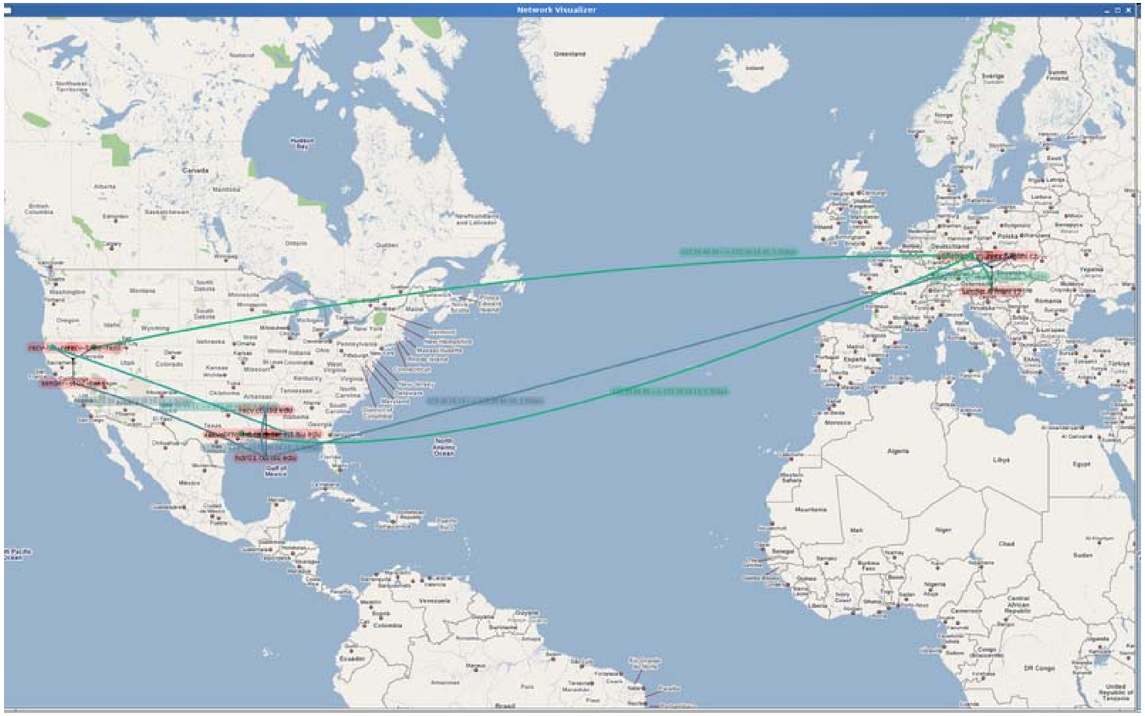
# Data Transfer Planning



Jakub Stoklasa

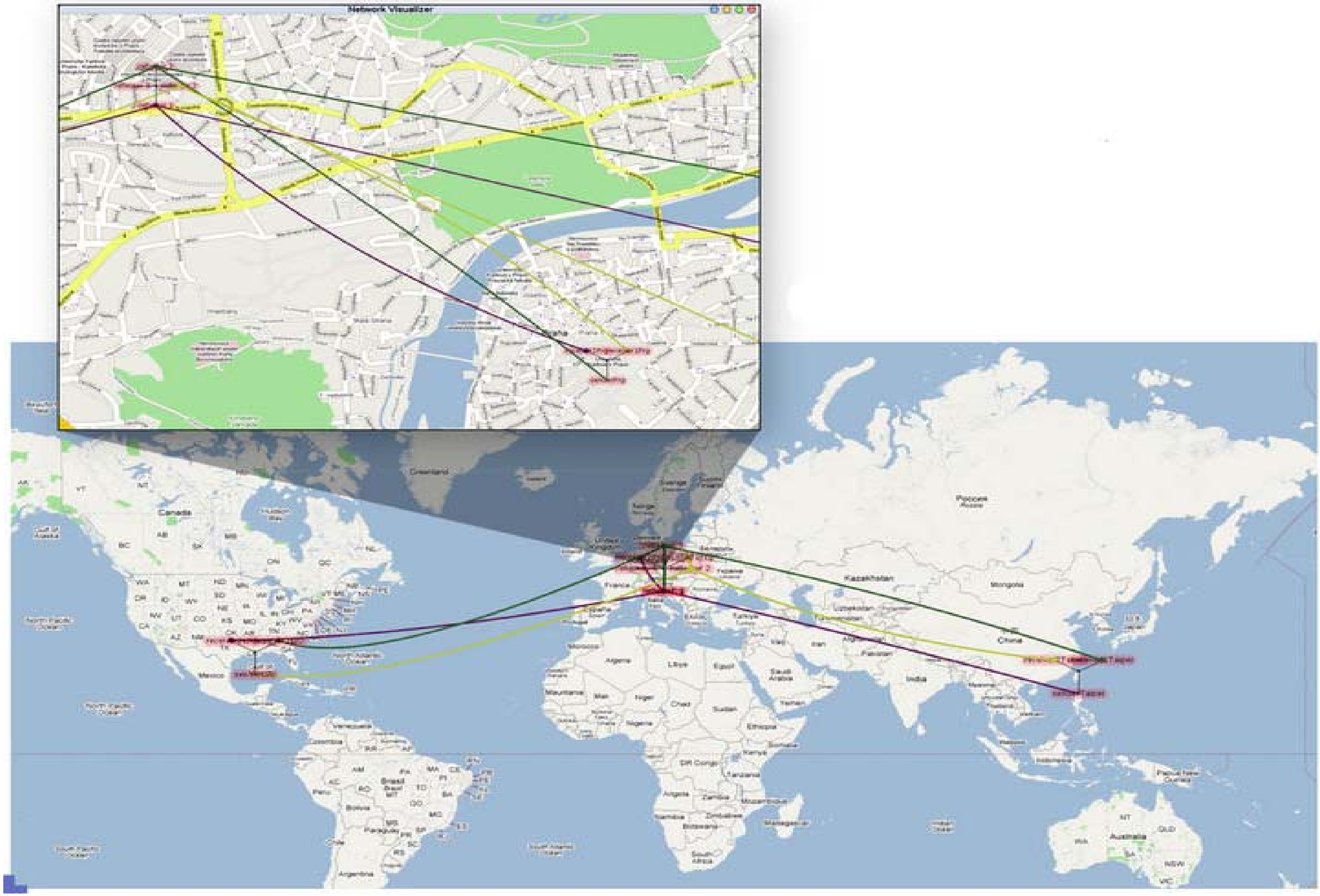
PV177 Laboratory of Advanced Network Technologies
April 14, 2010

# Contents

# CoUniverse

- Framework for building and self-organization of ad-hoc collaborative environments developed by Miloš Liška and Petr Holub (FI MUNI)
- Continuous adaptation on changing conditions based on built-in monitoring – re-planning from scratch on change
- Support for media streams with bitrate comparable to capacity of network links (e.g. HD video) – sophisticated scheduler needed
- Visualization of the environment for the users to make it understandable
- Uses a constraint based scheduler implemented in Java using a CHOCO solver library
- My work extends the original scheduler and adds some new features

# CoUniverse - GLIF2007 conference



# Problem Description

- Network Organization
- Media Applications
- Media Application Distributor
- Applications on Nodes Restrictions
- Stream Links
- Media Streams Planning Problem
- Network Topology Examples

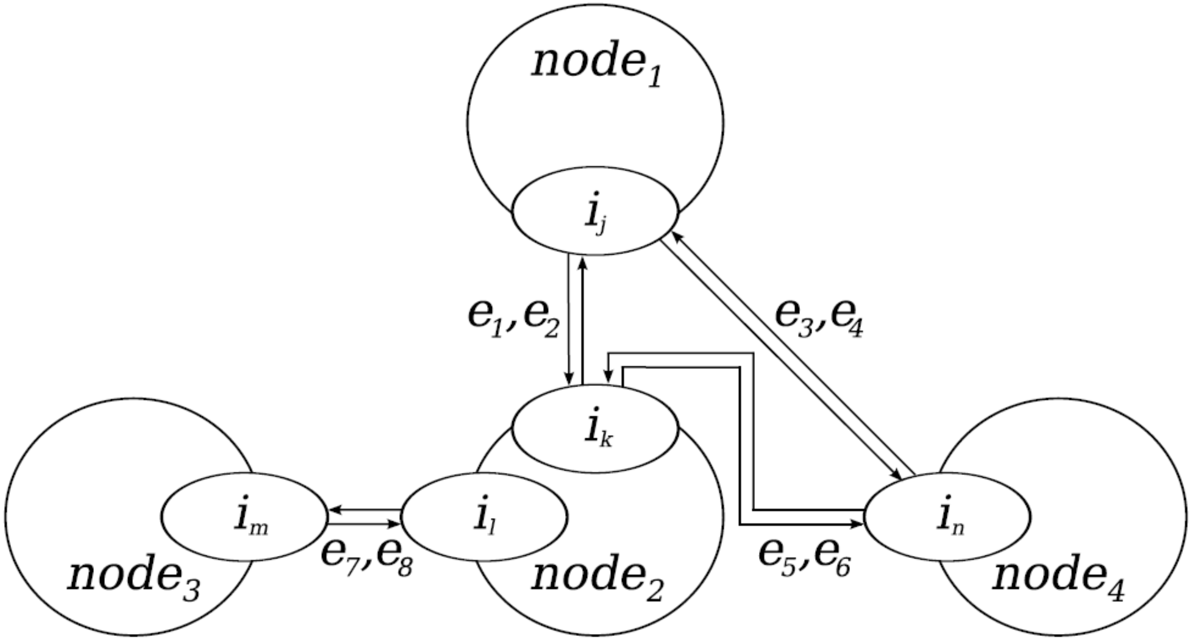
# Network Organization I

- Network represented as a graph $G = (V, E)$
  - Vertices → *Nodes*
  - Edges → *Links*

- *Sites* – geographical or logical (virtual) collocation of nodes
  – used to specify source for applications consuming data
- *Subnetworks* – separated parts of the network
- *Interfaces* – used to connect nodes within particular subnetworks
  – they describe a physical network infrastructure
- *Nodes* – configured with data processing applications
  – applications define capabilities of the node
- *Links* – full-mesh network topology between interfaces (of two different nodes) belonging to the same subnetwork

# Network Organization II

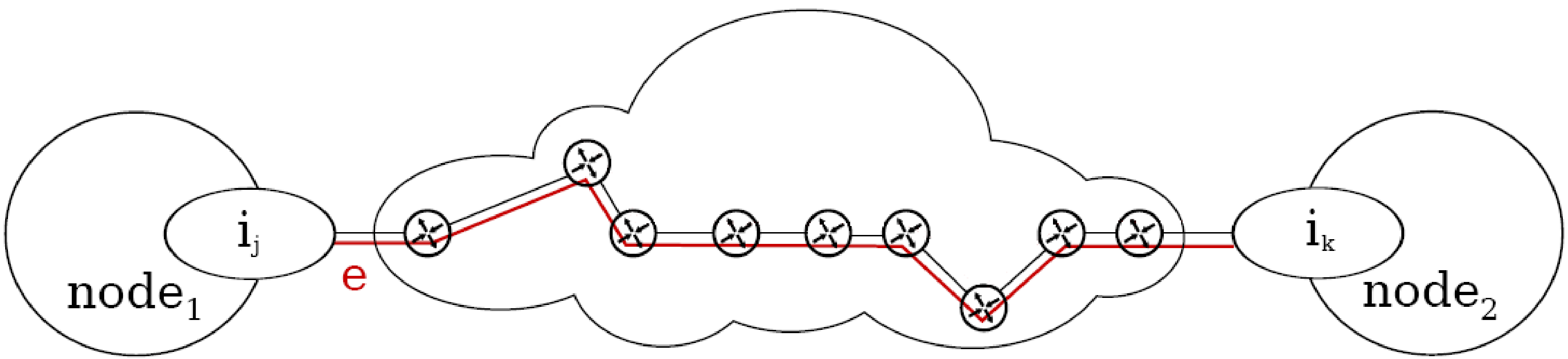
- $subnet(i_j) = subnet(i_k) = subnet(i_n) = net_1$
- $subnet(i_m) = subnet(i_l) = net_2$
- $node_2$ – "gateway" between $net_1$ and $net_2$




Nodes, interfaces and links example

# Network Organization III

- *Links* are comprehended as end-to-end links between node interfaces and thus they do not reflect the structure of real physical network topology
- Each *Link* in our model may be built using a number of physical network links, switches and routers




Physical network infrastructure vs. *Link*

# Media Applications

- Running on nodes
- Capable of producing and/or consuming data
- Communicate using *streams* of particular types
- *Stream* – abstract entity
  - – defined by its *Producer* and its *Stream Type*
- *Stream Type* – data (video) format (e.g. HDTV, HDV MPEG2)
  - – bandwidth, quality
- *Media Application Producer* / *Consumer*
  - capable of producing / consuming one or more *stream types*
  - e.g. UltraGrid, VideoLAN Client (VLC), Polycom device
- *Media Application Distributor*
  - special application for data distribution
  - receives a stream and proliferates it to other applications

# Media Application Distributor

- Application used for data distribution
- Consumes exactly one stream and is able to proliferate this (possibly transcoded) stream to more than one consumer (or other distributor/s)

- Transcoding type distributor (e.g. Active Element)
  - stream type of the input stream can be transcoded to some other stream type (dependent on distributor’s capability)
  - stream producer is always preserved!
- Reflector type distributor
  - no transcoding capabilities
  - only exact copies of the input stream can be distributed
  - used in previous version of constraint based scheduler

# Applications on Nodes Restrictions

- There are some restrictions on applications running on nodes for the input network:
- Each node has to run either

  just one $d \in D$ and no other application

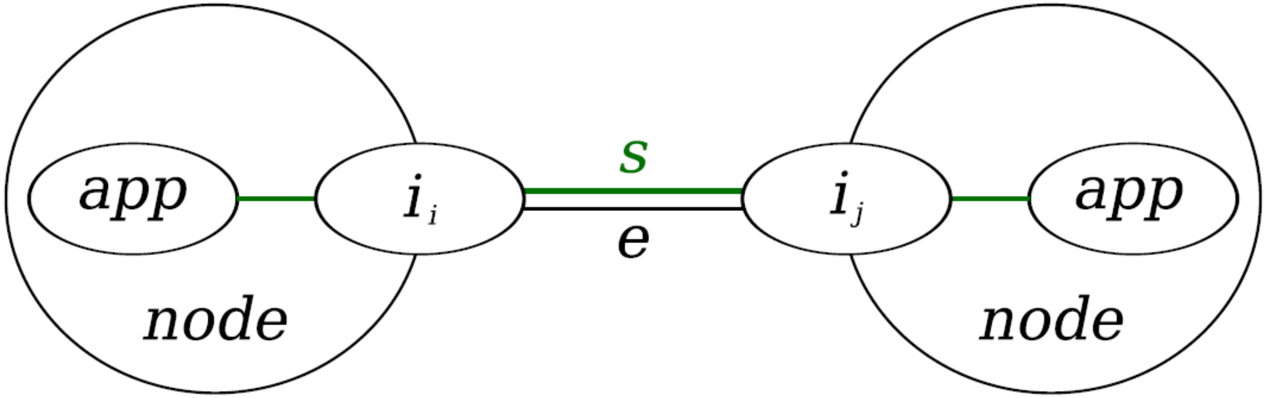
- Or

  $i$ producers and/or $j$ consumers where $(i + j) > 0$, $i \geq 0$, $j \geq 0$

- Two applications (consumers) processing a stream from the same source cannot run on the same node
- For example, two instances of UltraGrid consumers use fixed port number for addressing, thus cannot be listening for incoming media stream on a single node at the same time

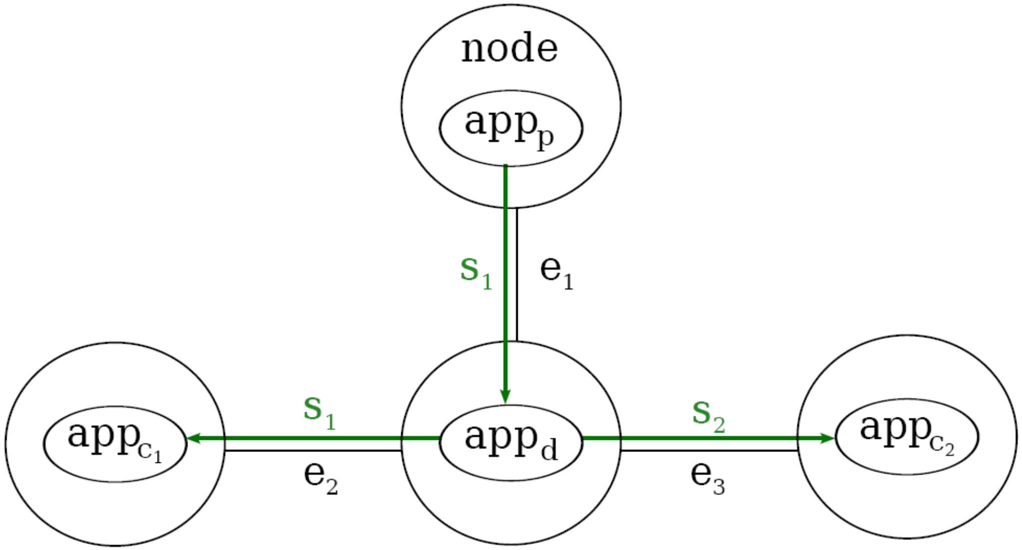
# Stream Links

- Abstraction of a fact that a stream is being transmitted over particular network link
- Basic entity to be scheduled in the proposed model
- Representation of stream link in proposed model: *sl*(*l*, *p*, *t*), where *l* is a network link, *p* is a producer and *t* is a stream type



# Media Streams Planning Problem

- The goal is to find such a set of media stream distribution trees (a forest) that all consumers are covered by producer/s from requested sites while satisfying all other conditions (distributors transcoding capabilities, links/interfaces capacities etc.)
- We want to optimize latency and/or quality of the solution

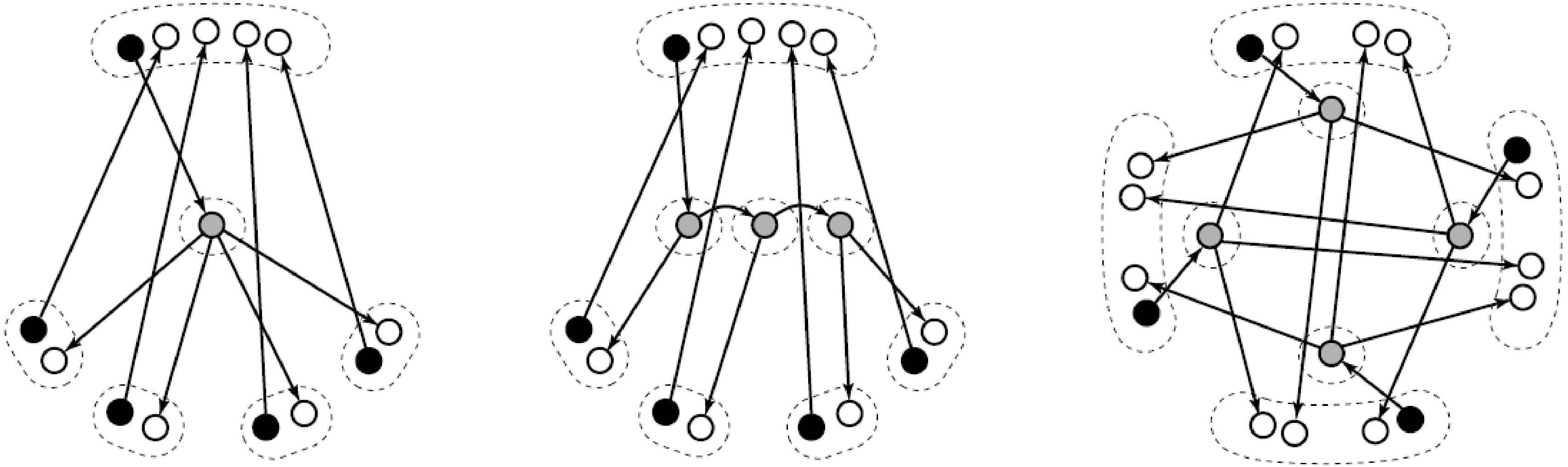



| stream | producer | type |
|---|---|---|
| $s_1$ | $app_p$ | A |
| $s_2$ | $app_p$ | B |

Media streams distribution tree example

# Network Topologies Examples

- Used for testing the simplified media streams planning problem
- (a) 1:n topology with a single distributor having sufficient capacity
- (b) 1:n topology with several distributors creating a distribution network
- (c) m:n full-mesh topology with a number of distributors



(a) (b) (c)

# Entities and Notation I

- *Nodes* (*V*)
  $\forall v \in V$:
  *site*(*v*) – particular site the node belongs to
  *interfaces*(*v*) – a set of interfaces belonging to the node *v*

- *Interfaces* (*I*)
  $\forall i \in I$:
  *subnet*(*i*) – just one subnet the interface belongs to
  *capacityI*(*i*) – capacity of the interface

- *Links* (*E*)
  – directed link $e = (i, j)$ where $i, j \in I$ are the terminal interfaces
  – such *e* exists iff $subnet(i) = subnet(j)$

# Entities and Notation II

- *Links* (cont.)

  $\forall e \in E$:

  $beginI(e)$ – beginning interface of the link $e$

  $endI(e)$ – ending interface of the link $e$

  $begin(e)$ – beginning node of the link $e$

  $end(e)$ – ending node of the link $e$

  *Note*: one interface can be shared by more links!

  $capacity(e)$ – capacity of the link $e$

  – determined by its interfaces (i.e. $\min(capacityI(i), capacityI(j))$) or further by the network monitoring

  $latency(e)$ – latency of the link $e$

  – determined solely by the network monitoring

# Entities and Notation III

- *Stream Types* ($T$)

  $\forall t \in T$:

  $bandwidth(t)$ – bandwidth needed to transfer a stream of type $t$

  $quality(t)$ – quality of a stream of type $t$

- *Producers* ($P$)

  $\forall p \in P$:

  $node(p)$ – parent node of the producer $p$

  $stream\_types(p)$ – a set of stream types the producer $p$ is able to produce

# Entities and Notation IV

- *Consumers* ($C$)

  $\forall c \in C$:

  *node*($c$) – parent node of the consumer $c$

  *stream_types*($c$) – a set of stream types the consumer $c$ is able to consume

  *requested_site*($c$) – a site from which the consumer $c$ wants to receive data and where appropriate producer is sought

- *Distributors* ($D$)

  $\forall d \in D$:

  *node*($d$) – parent node of the distributor $d$

  $transcode\_pairs(d) = \{(t_{in}, t_{out}) \mid t_{in}, t_{out} \in T\}$

# Entities and Notation V

Additional notation

- *"Belongs to node" notation*
  $\forall x \in P \cup C \cup D \cup I$ we write $x \in v$ meaning that producer/consumer/distributor/interface $x$ belongs to the node $v$

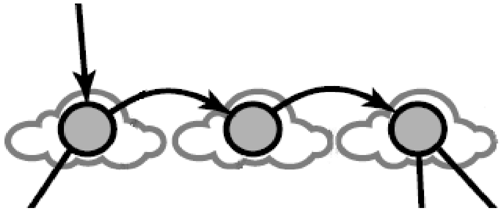
- *Sites* (*SI*)
  $$consumers(si) = \{\, c \mid c \in C \wedge requested\_site(c) = si \,\}$$

- *Distributors*
  $$transcode\_in(d) = \{\, t_{in} \mid (t_{in}, t_{out}) \in transcode\_pairs(d) \wedge t_{out} \in T \}$$
  $$transcode\_out(d) = \{\, t_{out} \mid (t_{in}, t_{out}) \in transcode\_pairs(d) \wedge t_{in} \in T \}$$
  $$transcode(d, t_{in}) = \{\, t_{out} \mid (t_{in}, t_{out}) \in transcode\_pairs(d) \wedge t_{out} \in T \}$$

# Entities and Notation VI

- *Producers*

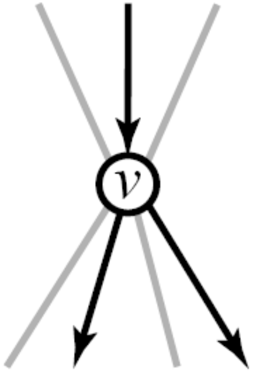
  *possible_types*(*p*) – a set of all types that streams of producer *p* can acquire in the given network configuration
  = *stream_types*(*p*) + their possible transcoding



- *Distributors & Producers*

$$indeg(d,p) = \sum_{\substack{(l \in E) \wedge \\ (node(d) = end(l))}} \; \sum_{\substack{t \in possible\_types(p) \cap \\ transcode\_in(d)}} sl(l,p,t)$$

$$outdeg(d,p) = \sum_{\substack{(l \in E) \wedge \\ (node(d) = begin(l))}} \; \sum_{\substack{t \in possible\_types(p) \cap \\ transcode\_out(d)}} sl(l,p,t)$$



# Pre-processing part I

- Eliminate inactive consumers (i.e. those where *requested_site*(*c*) = null)
- Eliminate producers from non-requested sites
- Generate a *possible_types*(*p*) set for each producer *p*

- Replace multi-input-type distributors by a set of *virtual distributors* (single-input type)
- Motivation example:

  *d*: *transcode_pairs*(*d*) = {(A, B), (A, C), (B, C)}


  – we are not able to distinguish whether the type C was transcoded from type A or type B

# Pre-processing part II

- Solution: we replace the original distributor by a set of virtual distributors on the particular node

$d$: $transcode\_pairs(d) = \{(A, B), (A, C), (B, C)\}$

$$\longrightarrow \begin{array}{l} d_1\text{: } transcode\_pairs(d_1) = \{(A, B), (A, C)\} \\ d_2\text{: } transcode\_pairs(d_2) = \{(B, C)\} \end{array}$$

- number of virtual distributors = $transcode\_in(d)$
- restriction on just one distributor on a node does not apply any more (it is restriction on the input network)
- only one of the virtual distributors can be active

- Network links elimination
  – helps to reduce number of network links and consequently the number of domain variables, thus making the problem smaller

# Network Links Elimination

- Eliminate edges that cannot be used for data transfer in our problem
- We will obtain significantly smaller number of stream links
- Stream link *sl*(*l*, *p*, *t*) will not be created for eliminated link *l*
- We want to find a set of edges

$$E_{\mathrm{elim}} = E \setminus \left\{ L_{\mathrm{cap}} \cup L_{\mathrm{c}} \cup L_{\mathrm{p}} \cup L_{\mathrm{site}} \right\}$$

where

$$L_{\mathrm{cap}} = \left\{ l \in E \mid capacity(l) < \min\left(\left\{ bandwidth(t) \mid t \in possible\_types(p) \wedge p \in P \right\}\right)\right\}$$

$$L_{\mathrm{c}} = \left\{ l \in E \mid p \notin begin(l) \wedge d \notin begin(l) \right\}$$

$$L_{\mathrm{p}} = \left\{ l \in E \mid c \notin end(l) \wedge d \notin end(l) \right\}$$

$$L_{\mathrm{site}} = \left\{ l \in E \mid site(begin(l)) = site(end(l)) \right\}$$

- In the following text we still denote a set of links as $E$ for sake of brevity but we treat it as $E_{\mathrm{elim}}$

# Constraint Model

- Domain Variables
- Capacity and Bandwidth
- Links to Node
- Links from Node
- Distributors
- Cycle Elimination
- Direct Links
- Optimization
- Constraint Satisfaction Problem
- Search Strategies

# Domain Variables

- We want to place requests (streams) to resources (network)
- Stream = producer + type

- Stream Links

$$X = \{\, sl(l, p, t) \mid l \in E, p \in P, t \in possible\_types(p) \,\}$$

- Boolean domain ($D$ = {0, 1})

$sl(l, p, t) = 0$ – stream from producer $p$ of type $t$ is not transmitted over link $l$

$sl(l, p, t) = 1$ – stream from producer $p$ of type $t$ is transmitted over link $l$

# Capacity and Bandwidth

- Capacity of each interface *i* must be sufficient to transfer all streams that are transmitted over links using the interface

$$\forall i \in I: \sum_{\substack{l \in E: \\ (i = beginI(l)) \\ \wedge (i = endI(l))}} \sum_{p \in P} \sum_{t \in possible\_types(p)} sl(l,p,t) \times bandwidth(t) \le capacityI(i) \qquad \textbf{(1)}$$

- Capacity of each link *l* must be sufficient to transfer all streams that are transmitted over the link

$$\forall l \in E: \sum_{p \in P} \sum_{t \in possible\_types(p)} sl(l,p,t) \times bandwidth(t) \le capacity(l) \qquad \textbf{(2)}$$

- Each link *l* must have sufficient capacity to transmit the stream of type *t* (redundant constraint)

$$\begin{gathered} \forall l \in E\ \forall p \in P\ \forall t \in possible\_types(p) \\ (bandwidth(t) > capacity(l)): \quad sl(l,p,t) = 0 \end{gathered} \qquad \textbf{(3)}$$

# Links to Node

- Each consumer *c* receives data by <u>just one</u> link carrying a stream of type *t* it is able to consume and which contains data produced by a producer *p* from the requested site

$$\forall c \in C: \sum_{\substack{(l \in E) \\ \wedge (c \in end(l))}} \sum_{\substack{(p \in P) \wedge \\ (site(node(p)) = \\ requested\_site(c))}} \sum_{\substack{t \in possible\_types(p) \cap \\ stream\_types(c)}} sl(l,p,t) = 1 \qquad \textbf{(4)}$$

- If there is neither an appropriate consumer nor an appropriate distributor at the ending node of the link *l*, this link cannot be used for transmitting the particular stream

$$\begin{aligned} &\forall l \in E\ \forall p \in P\ \forall t \in possible\_types(p) \\ &\quad ((\neg \exists c \in C\ ((c \in end(l)) \wedge (site(node(p)) = \\ &\qquad requested\_site(c)) \wedge (t \in stream\_types(c)))) \wedge \\ &\quad (\neg \exists d \in D\ ((d \in end(l)) \wedge (t \in transcode\_in(d)))) : \ sl(l,p,t) = 0 \end{aligned} \qquad \textbf{(5)}$$

# Links from Node I

- Each producer *p* sends data by <u>at most</u> one link out of all beginning at the node it is placed on

$$\forall p \in P: \sum_{(l \in E) \wedge (p \in begin(l))} \sum_{t \in stream\_types(p)} sl(l,p,t) \leq 1 \qquad \textbf{(6)}$$

- <u>At least one</u> producer from each requested site has to send data to the respective consumer/s (possibly distributed by distributors)

$$\forall si \in SI\ \exists c \in C\ (requested\_site(c) = si):$$
$$\sum_{(l \in E) \wedge (p \in begin(l))} \sum_{(p \in P) \wedge (site(node(p)) = si)} \sum_{t \in stream\_types(p)} sl(l,p,t) \geq 1 \qquad \textbf{(7)}$$

# Links from Node II

- If there is neither an appropriate producer nor an appropriate distributor at the beginning node of the link *l*, this link cannot be used for transmitting the particular stream

$$\forall l \in E\ \forall p \in P\ \forall t \in possible\_types(p)$$
$$((p \notin begin(l)) \vee (t \notin stream\_types(p)))$$
$$\wedge (\neg\exists d \in D\ ((d \in begin(l)) \wedge (t \in transcode\_out(d)))): \quad sl(s,l) = 0 \qquad \textbf{(8)}$$

# Distributors I

- Only one out of all (virtual!) distributors sharing a common parent node can be active

$$\forall v \in V \; \exists d', d'' \in D \; (d' \in v \wedge d'' \in v) : \sum_{(d \in D) \wedge (d \in v)} \sum_{p \in P} indeg(d, p) \leq 1 \quad \textbf{(9)}$$

  – applied only if number of distributors on a node is more than one
  – in case of one distributor (original) this constraint is not used as it would match the following constraint

- Each distributor *d* can be used for distribution of at most one input stream (i.e. it receives the data by one link at most)

$$\forall d \in D : \sum_{p \in P} indeg(d, p) \leq 1 \quad \textbf{(10)}$$

# Distributors II

- Data distribution constraints (distributors have to satisfy the following rules)

$$\forall d \in D\ \forall p \in P: \quad \begin{array}{l} \text{if indeg(d,p)} = 0 \text{ then outdeg(d,p)} = 0 \\ \text{if indeg(d,p)} = 1 \text{ then outdeg(d,p)} \geq 1 \end{array}$$

$$\forall d \in D\ \forall p \in P: \quad \text{if outdeg(d,p)} = 0 \text{ then indeg(d,p)} = 0$$

- Constraint for the first part of rules for *indeg*(*d*, *p*)

$$\forall d \in D\ \forall p \in P: \ indeg(d,p) \times outdeg(d,p) = outdeg(d,p) \qquad \textbf{(11)}$$

- Constraint for the second part of rules for *outdeg*(*d*, *p*)

$$\forall d \in D\ \forall p \in P: \ indeg(d,p) + outdeg(d,p) \neq 1 \qquad \textbf{(12)}$$

# Cycle Elimination

- To avoid cycles among the nodes with distributors, the cycle elimination constraint has to be used for each possible producer
- $n$ = number of nodes with distributors (i.e. number of distributors before generating the virtual distributors)

$$\forall p \in P \; \forall l \in E \; \forall k \; (2 \le k \le n) \; \forall i \; (1 \le i \le \tbinom{n}{k}):$$

$$\sum_{t \in possible\_types(p)} \; \sum_{\substack{j1, j2 \in C_{(n,k)}(i): \\ (v_{j1} = begin(l)) \wedge (v_{j2} = end(l)) \\ \wedge \, (1 \in sl(l,p,t))}} sl(l,p,t) < (k-1) \qquad \textbf{(13)}$$

- For each possible producer and for each $k$ smaller or equal than $n$, this constraint ensures that cycles among $k$ distributor nodes are prohibited

# Direct Links

- If there is more than one consumer for a particular site, data should be sent using some distributor and not directly from possible producer to respective consumers (redundant constraint)

$$\forall si \in SI \ \forall l \in E \ \forall p \in P \ \forall c \in consumers(si) \ \forall t \in possible\_types(p)$$
$$((\|consumers(si)\| > 1) \wedge (site(node(p)) = si) \qquad \textbf{(14)}$$
$$\wedge (p \in begin(l)) \wedge (c \in end(l))): \quad sl(l, p, t) = 0$$

- Problem: this constraint can eliminate some feasible solutions

# Optimization

- Latency minimization

$$\text{minimize} \sum_{l \in E} \sum_{p \in P} \sum_{t \in possible\_types(p)} sl(l, p, t) \times latency(l) \qquad \textbf{(15)}$$

- Quality maximization

$$\text{maximize} \sum_{l \in E} \sum_{p \in P} \sum_{t \in possible\_types(p)} sl(l, p, t) \times quality(t) \qquad \textbf{(16)}$$

# Constraint Satisfaction Problem I

- A set of domain variables – $X = \{ sl(l, p, t) \mid l \in E, p \in P, t \in possible\_types(p) \}$
- A domain of the variables – $D = \{0, 1\}$
- A set of essential constraints – $C = \{(1), (2), (4) – (13)\}$
- A set of all constraints (including the redundant ones) – $C^+ = \{(1) – (14)\}$
- A set of constraints for minimization problem – $C_{min}$ / $C^+_{min} = \{C / C^+ \cup (15)\}$
- A set of constraints for maximization problem – $C_{max}$ / $C^+_{max} = \{C / C^+ \cup (16)\}$
- A set of constraints for optimization problem – $C_{multi}$ / $C^+_{multi} = \{C / C^+ \cup (15), (16)\}$

# Constraint Satisfaction Problem II

- Consequently, we can define these corresponding CSPs:

$$P = (X, D, C)$$
$$P^+ = (X, D, C^+)$$
$$P_{\text{min}} = (X, D, C_{\text{min}}) \,/\, P^+_{\text{min}} = (X, D, C^+_{\text{min}})$$
$$P_{\text{max}} = (X, D, C_{\text{max}}) \,/\, P^+_{\text{max}} = (X, D, C^+_{\text{max}})$$
$$P_{\text{multi}} = (X, D, C_{\text{multi}}) \,/\, P^+_{\text{multi}} = (X, D, C^+_{\text{multi}})$$

- Each solution of described problems defines a forest where one tree in this forest corresponds to the data distribution of a set of streams from one producer to consumers
- We can have more distribution trees for one requested site (more than one producer can be active)

# Search Strategies

- Value ordering
  boolean variables – increasing (default), decreasing
- Variable ordering
  static:
  leftmost – simple linearization of *sl*(*l*, *p*, *t*) array over *l* first (outer loop) and then over *p* and its *t* (inner loop)
  rightmost – simple linearization of *sl*(*l*, *p*, *t*) array over *p* and its *t* first (outer loop) and then over *l* (inner loop)
  DFS – depth first search traversal from each possible producer
  BFS – breadth first search traversal from each possible producer
  dynamic:
  degree – based on the maximum number of constraints related with each variable

# Experimental Testing Proposal

Input instances configuration:

- Several different topologies (1:n, m:n, …)
- Consumers capable of receiving more than one stream type to be able to evaluate the maximization of the quality feature
- More sophisticated link latency values if possible to better evaluate the minimization of the latency feature

Experimental tests evaluation:

- Usage of different value and variable orderings
- Times needed to find a solution for different input instances and different types of CSP (optimization, all solutions, usage of the redundant constraints, …)
- Time to find only a first solution – appropriate for siginificantly large problems where finding optimal solution can take a long time

# Simplified Problem

- Solved by the original scheduler implemented by Miloš Liška and Petr Holub

- Precomputed matching of consumers and producers
- Only one producer from requested site can be active
- Selection of the producer is not unambiguous – there can be more suitable producers in the requested site, in such case the producer is chosen as a first match
- $X = \{sl(l, p) \mid l \in E, p \in P\}$ – significantly smaller problem
- $producer(c)$ – just one producer for the consumer $c$
- $consumers(p)$ – a set of consumers of the producer $p$

- Only reflector type distributors
- Only latency minimization as an objective function

# Evaluation of the Simplified P. I

Parameters of different topologies:

| topology-$\|SI_D\|$ | $\|V\|$ | $\|D\|$ | $\|E\|$ | $\|E_{\text{elim}}\|$ | $\|\mathcal{X}_{\text{elim}}\|$ | unass$(\mathcal{X}_{\text{elim}})$ | $\|(11)\|$ | $\|\Theta^+_{\text{elim}}\|$ | $F^o$ |
|---|---|---|---|---|---|---|---|---|---|
| *1:n-s-2* | 5 | 1 | 40 | 10 | 20 | 6 | 0 | 3 | 22 |
| *1:n-s-4* | 11 | 1 | 220 | 44 | 176 | 13 | 0 | 1 | 77 |
| *1:n-s-8* | 23 | 1 | 1,012 | 184 | 1,472 | 29 | 0 | 1 | 165 |
| *1:n-s-16* | 47 | 1 | 4,324 | 752 | 12,032 | 61 | 0 | 1 | 341 |
| *1:n-s-32* | 95 | 1 | 17,860 | 3,040 | 97,280 | 125 | 0 | 1 | 693 |
| *1:n-r-2* | 5 | 1 | 40 | 10 | 20 | 6 | 0 | 3 | 22 |
| *1:n-r-3* | 9 | 2 | 144 | 36 | 108 | 22 | 3 | 6 | 55 |
| *1:n-r-4* | 13 | 3 | 312 | 78 | 312 | 57 | 16 | 39 | 77 |
| *1:n-r-5* | 17 | 4 | 544 | 136 | 680 | 116 | 55 | 292 | 99 |
| *1:n-r-6* | 21 | 5 | 840 | 210 | 1,260 | 205 | 156 | 2505 | 121 |
| *1:n-r-7* | 25 | 6 | 1,200 | 300 | 2,100 | 300 | 399 | 24,306 | 143 |
| *m:n-2* | 6 | 2 | 60 | 18 | 36 | 14 | 2 | 7 | 22 |
| *m:n-3* | 12 | 3 | 264 | 60 | 180 | 45 | 12 | 6 | 99 |
| *m:n-4* | 20 | 4 | 760 | 140 | 560 | 112 | 44 | 24 | 176 |
| *m:n-5* | 30 | 5 | 1,740 | 270 | 1,350 | 225 | 130 | 120 | 275 |
| *m:n-6* | 42 | 6 | 3,444 | 462 | 2,772 | 396 | 342 | 720 | 396 |

# Evaluation of the Simplified P. II

## *1:n-s* topology [ms]

| $\|SI_D\|$ | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| first | 2.0±0.4 | 6.4±0.5 | 40.4±0.5 | 664±6 | 15,600±200 |
| min | 2.0±0.4 | 5.6±0.5 | 40.6±0.5 | 652±6 | 15,400±300 |

## *1:n-r* topology [ms]

| $\|SI_D\|$ | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| first | 2.0±0.4 | 4.0±0.4 | 9.6±0.5 | 20.6±0.5 | 40.0±0.4 | 81±2 |
| min | 2.0±0.4 | 4.8±0.4 | 13.6±0.5 | 39.2±0.5 | 240.8±0.4 | 3,050±70 |

## *m:n* topology [ms]

| $\|SI_D\|$ | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| first | 2.0±0.4 | 6.0±0.4 | 18±1 | 44.2±0.4 | 107.4±0.8 |
| min | 2.0±0.4 | 6.4±0.5 | 20.2±0.4 | 65.0±0.4 | 313±3 |

# Evaluation of the Simplified P. III

<u>Computational results for different variable and value ordering heuristics for selected topologies</u>

| | *1:n-s-32* | | *1:n-r-5* | | *m:n-r-4* | |
|---|---|---|---|---|---|---|
| | dec [ms] | inc [ms] | dec [ms] | inc [ms] | dec [ms] | inc [ms] |
| *leftmost* | 15,500±200 | 14,700±200 | 62,350±50 | 57,000±100 | 219,300±300 | 198,300±600 |
| *stream* | 15,400±200 | 14,700±200 | 510±20 | 475±1 | 88.4±0.5 | 84.8±0.4 |
| *dfs* | 16,400±80 | 16,200±300 | 304.6±0.8 | 283.4±0.8 | 51±0 | 48±0 |
| *degree* | 15,100±140 | 15,000±300 | 42.6±0.8 | 38.8±0.4 | 21.6±0.5 | 20.6±0.5 |

- All experimental tests results presented here have been taken from the *Data Transfer Planning with Tree Placement for Collaborative Environments* article written by Petr Holub, Miloš Liška and Hana Rudová. I thank for being able to use them for this presentation.