A Transport-Level Protocol Suite for Multimedia Multicast Communication over an SSM-enabled Internet

Roberto Canonico†, Laurent Mathy, David Hutchison

rcanonico@acm.org laurent@comp.lancs.ac.uk dh@comp.lancs.ac.uk

Lancaster University - Computing Department - Lancaster, LA1 4YR, UK

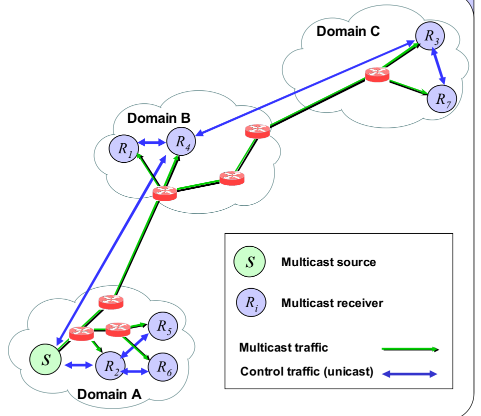
†Università di Napoli "Federico II" - Dipartimento di Informatica e Sistemistica - Napoli, Italy

LANCASTER UNIVERSITY

Computing Department

PIM-SSM for multicasting over the Internet: implications for end-to-end protocols

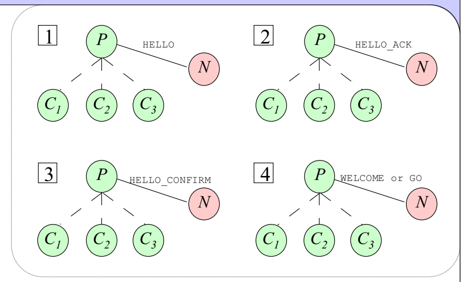
- PIM-SSM introduces a new model of multicasting on the Internet, suitable for applications with few well-known sources
  - PIM-SSM currently defined by I-D `draft-holbrook-ssm-arch-00.txt`
  - IANA has allocated the 232 / 8 address block for exclusive use of Source-Specific Multicast
- PIM-SSM realizes components of the EXPRESS multicast service model based on source-specific *channels*
- When PIM-SSM is adopted, there is no way for a receiver to multicast a packet to all other receivers of an (S,G) PIM-SSM channel
- This, indeed, is one of the benefits of the SSM model, since it protects receivers from unwanted traffic
- However, many transport-level protocols rely on a "multicast back-channel"
  E.g. used by receivers to send control messages "up-stream" to the source and all other group receivers
- **Our solution:** to build a "parallel" infrastructure for control purposes by using Application Level Multicasting
  This infrastructure relies on unicast (i.e point-to-point) connections among end-systems to form a spanning tree rooted at the channel source
- **Our constraints:**
  - Since we work in end-systems, we assume no knowledge of network topology and routing
  - Since we want to limit the complexity of our algorithms, we assume only "partial" knowledge of group membership



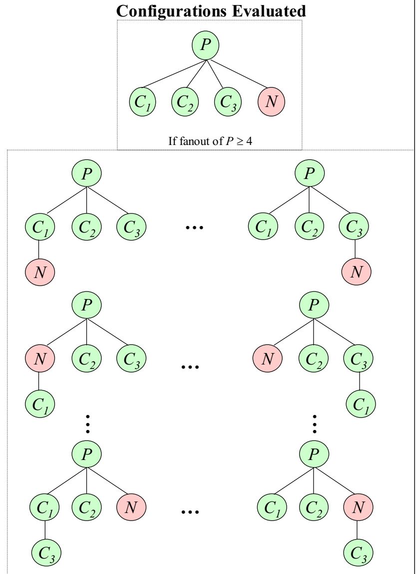
The Control Tree Building algorithm

- **Goal:** to build a spanning tree rooted at the channel sender S and connecting all receivers R_i exploiting only "partial" info on the group membership and network topology
- **Utilisation:** as a support infrastructure for all those control protocols that need a "multicast back-channel"
  - In all these cases, messages will flow upstream from receivers towards the sender S
  - In some cases, messages could be aggregated at intermediate nodes, to avoid source implosion
- **Typical applications:** receivers counting, group integrity management, nack transmission, local repairs, ...
- **Basic algorithm:** The spanning tree is built as receivers join the channel through a distributed algorithm that starts at the tree root
  - Each node in the tree accommodates up to *fanout* children (*fanout* ≥ 1)
  - A receiver N finds its place in the tree by contacting a candidate parent P, starting from the *tree root* S (message `HELLO`)
  - P sends back to N the list of its own children C_i (message `HELLO_ACK`)
  - N measures its distance from P and all C_is and sends this info to P (message `HELLO_CONFIRM`)
  - P finds a place for N, by evaluating all possible configurations having P as parent and $\{C_i\}\cup\{N\}$ as children
  - One of the following three cases can occur:
    - P accepts N as child and it does not need to move one of the previous children C_is (message `WELCOME` to N)
    - P accepts N as child but it needs to move its child C_j as a child of C_k (message `WELCOME` to N, `GO(`C_k`)` to C_j)
    - P sends N to its child C_k (message `GO(`C_k`)` to N) and the algorithm continues until N receives a message `WELCOME`
  - To evaluate all the possible configurations, a **score function** is used to measure "how good" a configuration is
  - Let NC = no. of children of P, it is:

$$\text{No. of configurations to be evaluated} = \begin{cases} NC^*(NC+1)+1, & \text{if fanout}(P) > NC \\ NC^*(NC+1), & \text{if fanout}(P) = NC \end{cases}$$

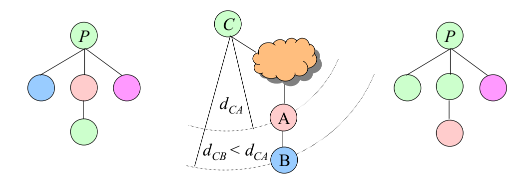


Configurations Evaluated



- **Variant:** ***hierarchical organisation of receivers in "domains"***
  - **Motivation:** receivers are usually "clustered"
  - **Goal:** the tree is built in a way that receivers of the same domain are all children of the same node called "domain root"
  - **Assumption:** Each receiver knows its own *domainID* (e.g. `IP-address && netmask`)
  - Receivers declare their own *domainID* when connecting to a new candidate parent
  - The tree root maintains a table of *domain roots*
  - New receivers are redirected to existing *domain roots*
    If a *domain root* does not exist for a domain, the new comer is elected *domain root* of its domain
- **Notes on the score function**
  - The score function determines the shape of the tree
  - What is the "best" shape for a control tree?
  - We also impose three constraints that must be satisfied for a configuration to be "acceptable"
  - **Rule 1:** all the nodes of a domain should remain "clustered"
  - **Rule 2:** the tree should grow outward from the tree root (or the domain root)
  - **Rule 3:** domain root should be as high as possible in the tree (i.e. as close as possible to the tree root)
- [*] **Note:** if *domainID*(A) == *domainID*(B) then C is their common domain root, otherwise C is the tree root (S)

Note: *In the pictures below, colours are used to identify the domain each node belongs to.*



Rule no.1 discards configurations like this. *Rule no.2 discards configurations like this (see note [*]).* *Rule no.3 discards configurations like this.*

A Transport-Level Protocol Suite for Multimedia Multicast Communication over an SSM-enabled Internet

Roberto Canonico†, Laurent Mathy, David Hutchison

rcanonico@acm.org laurent@comp.lancs.ac.uk dh@comp.lancs.ac.uk

Lancaster University - Computing Department - Lancaster, LA1 4YR, UK

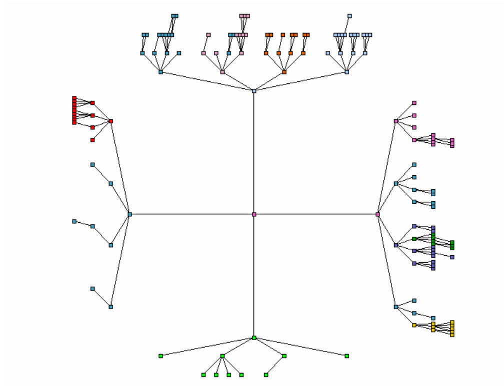
†Università di Napoli "Federico II" - Dipartimento di Informatica e Sistemistica - Napoli, Italy

LANCASTER UNIVERSITY

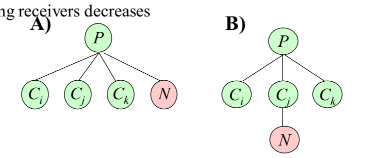
Computing Department

Evaluation

- The Tree Building Protocol has been tested with the *ns-2* simulator, by generating a random topology consisting of:
  - a network "core", composed of 25 nodes
  - 15 stub networks, each composed of 10 nodes
- 150 protocol entities were placed in end-systems connected to stub networks
- 100 simulation runs were executed in each condition, to calculate average and maximum performance indices
- To characterise the trees built with our algorithm we plot the following performance indices against group size (i.e. no. of receivers)
  - mean and maximum distance of a node from its parent
  - mean and maximum distance of a node from its domain root
  - mean and maximum distance of a node from the tree root
  - mean and maximum link stress (no. of copies of the same packet on a link)
  - delay penalty, i.e. the ratio between the delay along the tree and the delay along a direct connection to the sender
- **Observations:**
  - the tree built by our algorithm depends on the order of arrival of receivers
  - when the "density" of receivers increases, the average distance among receivers decreases



A typical tree built by our algorithm
(150 receivers densely spread - fanout = 4)
Colours are used to identify domains.
Graph produced with the Otter visualisation tool available at
`http://www.caida.org/tools/visualization/otter/`



Cost function used to obtain these results:

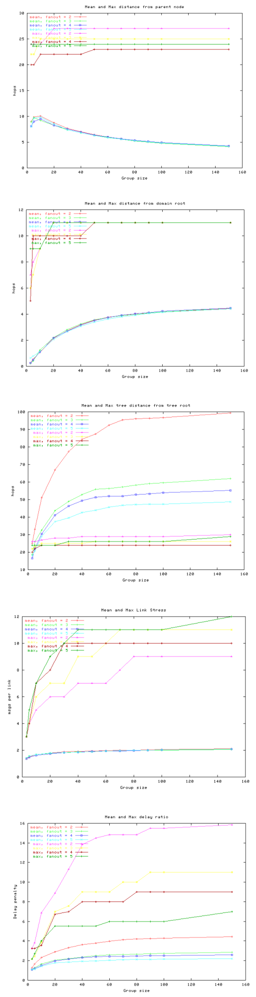
$$score = \begin{cases} \text{distance}(P,N) & \text{if case } A) \\ \text{distance}(C_i,N) & \text{if case } B) \end{cases}$$

The configuration with the minimum score is chosen

However, to slightly favour a growth of the tree in breadth, rather than in depth, we apply two corrective actions:

1) We discard configurations of the type B when their score is no less than a given fraction of the score of configuration of type A

2) When more configurations exhibit a similar score, the configuration is chosen randomly among them



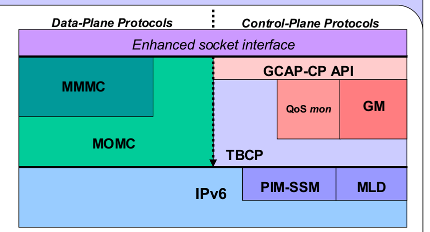
Protocol implementation in the GCAP protocol suite

Overview of the GCAP Project

- The work presented here is being developed within the European Project GCAP, in collaboration with the LIP6 Laboratory of the University "Pierre et Marie Curie" of Paris. GCAP is funded by the European Commission under the Framework V IST Programme.
- GCAP's goal is to design, implement and test a suite of transport-layer protocols to support multimedia multicast applications
- GCAP assumes as underlying network-layer service the one provided by an IPv6 network enabled with multicast PIM-SSM routers
- GCAP Protocol Entities will be deployed in end-systems and in active edge devices
- GCAP Protocol Entities deployed in edge devices could behave as domain roots for all the receivers in their domain

The GCAP Protocol Suite

- The GCAP Protocol Suite can be divided into Data-Plane Protocols and Control-Plane Protocols
- Data-Plane Protocols provide end-to-end transport of multimedia data units, with Partial Order / Partial Reliability, according to a *Fully Programmable Transport Protocol* model
- Control Plane Protocols provide support functionality for multimedia multiparty applications, such as Receivers Counting, Group Management, QoS monitoring, Reliability, …
- All GCAP protocols are implemented in Java as user-level libraries



The GCAP Protocol Suite

Data-Plane Protocols:
MMMC: *Multimedia Multicast Transport Protocol*
MOMC: *Monomedia Multicast Transport Protocol*
Control-Plane Protocols:
TBCP: *Tree Building Control Protocol*
GM: *Group Management Protocol*
QoS-mon: *QoS monitoring Protocols*

Further investigation and future work

- Testing of different cost functions for intra-domain placement and domain-roots placement
- Design of end-to-end control protocols on top of the Control Tree
- Refinement of leave mechanisms
- Definition of tree reshaping mechanisms, triggered by receivers
- Usage of PIM-SSM in IPv6 networks – Extension of the Multicast Listener Discovery protocol (MLD) to support IGMPv3 source-specific operations
- Java implementation by the end of the year