Transport-level Protocol Coordination in Cluster-to-Cluster Applications

David E. Ott and Ketan Mayer-Patel
University of North Carolina at Chapel Hill
{ott,kmp}@cs.unc.edu

Abstract

Future Internet applications will increasingly use multiple communications and computing devices in a distributed fashion. In this paper, we identify an emerging and important application class comprised of a set of processes on a cluster of devices communicating to a remote set of processes on another cluster of devices across a common intermediary Internet path. We define these types of applications as *cluster-to-cluster applications*, or *C-to-C applications*. The networking requirements of C-to-C applications present unique challenges that current transport-level protocols fail to address. In particular, these applications require aggregate measurement of network conditions across all associated flows and coordinated transport-level protocol behavior. A *Coordination Protocol* (CP) is proposed which allows a C-to-C application to coordinate flow behavior in the face of changing network conditions. CP provides cluster endpoints with a consistent view of network conditions, as well as cluster membership and bandwidth usage information. An application may use CP to define and implement a coordination scheme supporting particular flow priorities and other objectives.

1 Introduction

Future Internet applications will increasingly make use of multiple communication and computing devices in a distributed fashion. Examples of these applications include distributed sensor arrays, tele-immersion [9], computer-supported collaborative workspaces (CSCW) [4], ubiquitous computing environments [13], and complex multi-stream, multimedia presentations [16]. In many such applications, no one device or computer produces or manages all of the data streams transmitted. Instead, the endpoints of communication are collections of devices. We call applications of this type *cluster-to-cluster applications*, or *C-to-C applications*.

In a C-to-C application, a set of processes distributed on a cluster of devices or computers communicates with another set of processes on a remote cluster of devices or computers. For example, a cluster may be comprised of a collection of capture devices (e.g., video cameras, microphones, etc.), each of which produces a stream of data. The other cluster might be a collection of display devices (e.g., digital light projectors, speakers, head- mounted displays, etc.) and computers that archive data in various ways. Processes are distributed on each device cluster to manage data streams and control the application.

C-to-C applications share two important properties. First, communication between endpoints on the same local cluster takes place with minimal delay and loss. We can make this assertion because the local intranet supporting the cluster can be provisioned to comfortably support application traffic loads. In other words, the local networking environment often will be engineered and implemented with the C-to-C application in mind. Second, while no two flows within the application share the exact same end-to-end path, all flows share a common Internet path between clusters. This shared common path represents the majority of the traversal path between endpoints on different clusters, and the region in which network congestion will most likely affect C-to-C application performance. Although the path between clusters is not guaranteed to be the same for all flows (or even all packets within a flow), in general we can expect that network conditions between clusters will be similar.

Different flows within a C-to-C application may use more than one transport-level protocol to accomplish their communication objectives. Streamed audio and video data, for example, may use UDP or a UDP- based protocol like RTP, while control information is exchanged using TCP to ensure reliability. Application-specific protocols which handle media encoding, flow control, reliability, or congestion control in specialized ways may also be used.

A fundamental problem with current transport-level protocols within the C-to-C application context, however, is that they lack coordination. Application streams share a common intermediary path between clusters, and yet operate in isolation from one another. As a result, flows may compete with one another when network resources become limited, instead of cooperating to use available bandwidth in application-controlled ways. For example, an application may wish to give certain streams priority over others or stipulate that different proportions of available bandwidth be used by specific streams. We are interested in C-to-C applications with rich semantic relationships between flows that can be exploited to improve application performance.

In this position paper, we postulate that the C-to-C application architecture can significantly benefit from network mechanisms that allow transport-level protocol coordination of separate, but semantically related, flows of data. In the following sections of this paper we will:

- Motivate the C-to-C application architecture with a

Figure 1: The Office of the Future.

specific example.

- Identify a number of networking challenges posed by such an application.
- Provide a specific proposal for a transport-level coordination mechanism.
- Defend the design decisions made in our proposed mechanism, and discuss related work.

2 A Motivating Example

This section describes a specific application which motivates the need for transport-level protocol coordination and illustrates the networking challenges faced by C-to-C applications. The application is the *Office of the Future*. The Office of the Future was conceived by Fuchs et al., and their experimental prototype of the application is described further in [9].

In the Office of the Future, tens of digital light projectors are used to make almost every surface of an office (walls, desktops, etc.) a display surface. Similarly, tens of video cameras are used to capture the office environment from a number of different angles. At real time rates, the video streams are used as input to stereo correlation algorithms to extract 3D geometry information. Audio is also captured from a set of microphones. The video streams, geometry information, and audio streams are all transmitted to a remote Office of the Future environment. At the remote environment, the video and audio streams are warped using both local and remote geometry information and stereo views are mapped to the light projectors. Audio is spatialized and sent to a set of speakers. Users within each Office of the Future environment wear shutter glasses that are coordinated with the light projectors.

The result is an immersive 3D experience in which the walls of one office environment essentially disappear to reveal the remote environment and provide a tele-immersive collaborative space for the participants. Furthermore, synthetic 3D models may be rendered and incorporated into both display environments as part of the shared, collaborative experience. Figure 1 is an artistic illustration of the application.

We see the Office of the Future as a concrete vision of a C-to-C application that progresses far beyond today's relatively simple video applications. The Office of the Future uses scores of media streams which must be manipulated in complex ways. The computational complexity of the application requires the use of several computational resources (possibly including specialized graphics hardware).

The Office of the Future is a good example of a C-to-C application because the endpoints of the application are collections of devices. Two similarly equipped offices must exchange myriad data streams. Any specific data stream is transmitted from a specific device in one environment to one or more specific devices in the remote environment. Within the strict definition of end-to-end used by current protocols, few if any of these streams will share a complete path. If we relax the definition of end-to-end, however, we can see that all of the data streams will span a common shared communication path between the local networking environments of each Office of the Future.

The local network environments are not likely to be the source of congestion, loss, or other dynamic network conditions because these environments are within the control of the local user and can be provisioned to support the Office of the Future application. The shared path between two Office of the Future environments, however, is not under local control and thus will be the source of dynamic network conditions.

Current transport-level protocols fail to address the complex needs of an application like the Office of the Future. For example, this application requires dynamic interstream prioritization. Beyond just video information, the Office of the Future uses a number of other media types including 3D models and spatialized audio. Because these media types are integrated into a single immersive display environment, user interaction with any given media type may have implications for how other media types are encoded, transmitted, and displayed. For example, the orientation and position of the user's head indicates a region of interest within the display. Media streams that are displayed within that region of interest should receive a larger share of available bandwidth and be displayed at higher resolutions and frame rates than media streams that are outside the region of interest. When congestion occurs, lower priority streams should react more strongly than higher priority streams. In this way, appropriate aggregate behavior is achieved and application-level tradeoffs are exploited.

3 Networking Challenges

To generalize the networking challenges faced by complex C-to-C applications like the Office of the Future, we first develop a generic model for C-to-C applications and subsequently characterize the networking requirements associated with this model.

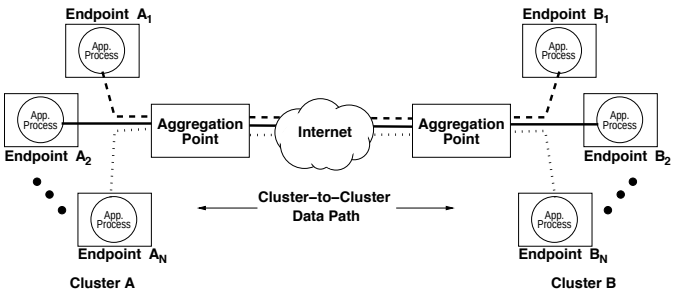

Figure 2: C-to-C application model.

3.1 C-To-C Application Model

We model a generic C-to-C application as two sets of processes executing on two sets of communication or computing devices. Figure 2 illustrates this model.

A *cluster* is comprised of any number of *endpoints* and a single *aggregation point*, or *AP*. Each endpoint represents a process on some *endpoint host* (typically a networked computer) that sends and/or receives data from another endpoint belonging to a remote cluster. The AP functions as a gateway node connecting cluster endpoints to the Internet. The common traversal path between aggregation points is known as the *cluster-to-cluster data path*.

Endpoints within the same cluster are connected by an intranet provisioned with enough bandwidth to comfortably support the communication needs of the cluster. In general, latency between endpoints on the same cluster is small compared to latency between endpoints on different clusters. Our overall objective is to coordinate endpoint flows across the cluster-to-cluster data path where available bandwidth is uncertain and constantly changing.

3.2 Networking Requirements

A useful metaphor for visualizing the networking requirements of C-to-C applications is to view the communication between clusters as a rope with frayed ends. The rope represents the aggregate data flows between clusters. Each strand represents one particular transport-level stream. At the ends of the rope, the strands may not share the same path. In the middle of the rope, however, the strands come together to form a single aggregate object. While each strand is a separate entity, they share a common fate and purpose when braided together as a rope.

With this metaphor in mind, we identify several important networking requirements of C-to-C applications:

- **Global measurements of congestion, delay, and loss.** Although each stream of a C-to-C application is an independent transport-level flow of data, they are semantically related as part of a single distributed application. As such, network events such as congestion, delay, and loss should be measured and reported for the flows as an aggregate.

- **Preserved end-to-end semantics.** The specific transport-level protocol (i.e., TCP, UDP, RTP, RAP, etc.) that is used by each flow will be specific to the communication requirements of the data within the flow and the role it plays within the larger application. Thus, each transport-level protocol used should still maintain the appropriate end-to-end semantics and mechanisms. For example, if a data flow contains control information that requires in-order, reliable delivery, then the transport-level protocol used to deliver this specific flow (e.g., TCP) should provide these services on an end-to-end basis. Thus, although the flow is part of an aggregate set of flows, it should still maintain end-to-end mechanisms as appropriate.

- **A coordinated view of current network conditions.** Even though each flow maintains its own end-to-end semantics, the flows should receive a coordinated view of current network conditions like available bandwidth and global measures of aggregate delay, loss, and congestion. We need to separate the adaptive dynamic behavior of each transport-level protocol which depends on end-to-end semantics of individual streams, from the mechanisms used to measure current network conditions which should be measured globally across all streams of the application.

- **Interstream relative bandwidth measurements.** Individual streams within the C-to-C application may require knowledge about their bandwidth usage relative to the other streams of the same application. This knowledge can be used to determine the appropriate bandwidth level of a particular stream given application-level knowledge about interstream relationships. For example, an application may want to establish a relationship between two separate flows of data such that one flow consumes twice as much bandwidth as the other.

- **Deployability** Finally, the mechanisms used to provide C-to-C applications with transport-level protocol coordination need to be reasonably deployable.

In the following section, we outline a design for a mechanism that meets these requirements.

4 The Coordination Protocol

Our approach to this problem is to propose a new protocol layer between the network layer (IP) and transport layer (TCP, UDP, etc.) that addresses the need for coordination in C-to-C application contexts. We call this protocol the *Coordination Protocol (CP)*. The coordination function provided by CP is transport protocol independent. At the same time, CP is distinct from network-layer protocols like IP that play a more fundamental role in routing a packet to its destination.

CP works by attaching probe information to packets transmitted from one cluster to another. As the information is acknowledged and returned by packets of the remote cluster, a picture of current network conditions is formed

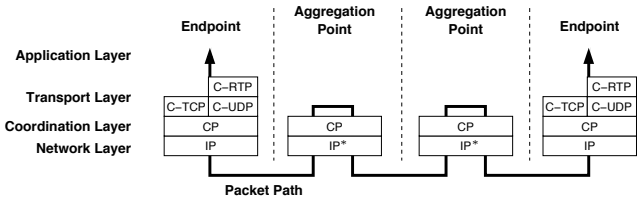

Figure 3: CP network architecture.

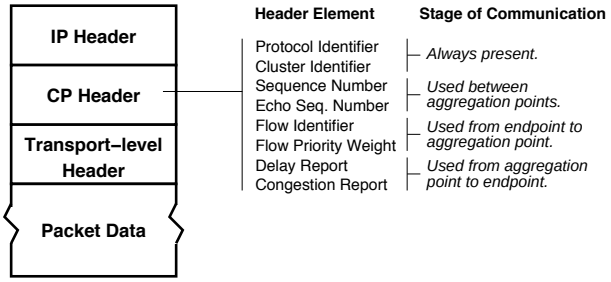

Figure 4: CP packet structure.

and shared among endpoints within the local cluster. A consistent view of network conditions across flows follows from the fact that the same information is shared among all endpoints.

Figure 3 shows our proposed network architecture. CP will be implemented between the network layer (IP) and the transport layer (TCP/UDP). We call this layer the *coordination layer*. The coordination layer will exist on each device participating in the C-to-C application, as well as on the two aggregation points on either end of the cluster-to-cluster data path.

At the endpoints, CP can be implemented on top of IP in a straightforward manner much as TCP and UDP are implemented on top of IP. At the aggregation points, however, CP must be part of the forwarding path for any CP packets. As a result, IP forwarding behavior must be modified at the aggregation points to pass CP packets through CP before being forwarded. We acknowledge this departure from standard IP behavior by labeling the IP layer at the AP's in Figure 3 as IP*.

Transport-level protocols will be built on top of CP in the same manner that TCP is built on top of IP. CP will provide these transport-level protocols with a consistent view of network conditions. These transport-level protocols will in turn use this information, along with various configuration parameters, to determine a data transmission rate and related send characteristics. In Figure 3, we show several possible transport-level protocols (C-TCP, C-UDP, and C-RTP) which are meant to represent coordinated counterparts to existing protocols.

4.1 Basic Operation

Figure 4 shows a CP data packet. CP encapsulates transport-level packets by prepending a small header. In turn IP will encapsulate CP packets and the protocol header will indicate that CP is being used. Each CP header will contain an identifier associating the packet with a particular C-to-C application. The remaining contents vary according to the changing role played by the CP header as it traverses the network path from source endpoint to destination endpoint.

- **As packets originate from source endpoints.**
  The CP header is used to communicate requests to the local AP. For instance, an endpoint may request to join or leave a cluster, have a membership or bandwidth usage report issued, or post a message to all endpoints within the local cluster.

- **As packets arrive at the local AP.**
  CP will process the information arriving in CP headers. Part of the CP header will then be overwritten, allowing the AP to communicate congestion probe information with the remote AP.

- **As packets arrive at the remote AP.**
  The CP header is processed and used to detect network conditions. Again, part of the CP header is overwritten to communicate network condition information with the remote endpoint.

- **As packets arrive at the destination endpoint.**
  CP processes network condition information from the CP header and passes it on to the transport-level protocol and the application.

4.2 Detecting Network Conditions

A primary function of CP is to measure congestion and delay along the cluster-to-cluster data path. To accomplish this objective, each packet passing from one AP to another will have two numbers inserted into its CP header. The first number will be a sequence number that increases monotonically for every packet sent. The second number will be an echo of the last sequence number received from the other AP.

By recording the time when a new sequence number is sent and the time when its echo is received, the AP can approximate the round trip time and infer network delay. Similarly, missing sequence numbers can be used to detect packet loss and infer congestion.

Because sequence numbers in the CP header do not have any transport-level function, CP can use whatever packet is being transmitted next to carry this information as long as the packet is part of a flow associated with the C-to-C application. Since the packets of multiple flows are available for this purpose, this mechanism can be used for fine-grained detection of network conditions along the cluster-to-cluster data path.

4.3 Transport-level Interface

An important aspect of CP is the design of its interface with transport-level protocols. CP provides transport-level protocols with information about network conditions, includ-

ing congestion and delay measurements. These measurements are delivered whenever packets pass from CP to the transport-level protocol through this interface.

Variants of existing transport-level protocols will be developed on top of this interface. For example, a coordinated version of TCP (C-TCP) will consider acknowledgements only as an indicator of successful transfer. Congestion detection will be relegated entirely to CP.

Because an application will need to provide CP-related information to the transport-level protocol, some aspects of the transport-level interface will be exposed to the networking API (i.e., the socket layer). For instance, an application will need to supply a cluster identifier that associates the flow with a given C-to-C application. In the reverse direction, some transport-level protocols will make delay, jitter, loss, or other CP-derived information available to the application layer.

4.4 Cluster Membership Services

An AP maintains a cluster membership table for each C-to-C application. This table is created when a CP packet for a C-to-C application with no current table is received and is maintained as soft state. Endpoints are dynamically added and deleted from this table. Storing the table as soft state avoids explicit configuration and makes the mechanism lightweight and robust

A cluster membership table performs two important functions. First, it maintains a list of known cluster endpoints. This allows the AP to broadcast network condition information to all C-to-C application participants. Endpoints may also request a membership report from the AP to receive a current listing of other endpoints. This list may be used in application-specific ways, for example to communicate point-to-point with cluster participants or to track cluster size.

The second function is that of bandwidth monitoring. An AP will monitor bandwidth usage for each cluster endpoint and use the cluster membership table to store resulting statistics. An endpoint may request a bandwidth report to obtain information on bandwidth usage among other endpoints, as well as aggregate statistics on the cluster as a whole. This information can be used in application-specific ways to configure and manage flow coordination dynamically.

5 Design Rationale

5.1 Benefits of CP

We believe the benefits of our approach to be substantial and include:

- **Locally deployable.**
  CP requires changes to protocol stacks within application clusters only. No changes are required of forwarding nodes along the Internet path between clusters. Furthermore, cluster endpoints can coexist with non-CP communication and computing devices.

- **Available to all transport protocols.**
  A consistent view of network conditions is accessible by all flow endpoints, regardless of the transport-level protocol used by a given flow.

- **Flexible.**
  Our approach does not dictate how an application handles flow coordination. Nor does it enforce coordination schemes through external traffic shaping or packet scheduling at switching points. Instead, flow endpoints are informed of network conditions and adjust in application-defined ways.

- **Dynamic membership and configuration.**
  The number of endpoints within a cluster may change dynamically, along with the traffic characteristics of any given flow endpoint.

- **Sensitive to router performance issues.**
  While the CP architecture does require per-flow and per-cluster state to be maintained at the aggregation points, the work involved is limited to simple accounting operations and does not involve packet scheduling, queue management, or other complex processing.

It is important to note that CP is only one possible implementation of a mechanism to satisfy the networking requirements of C-to-C applications outlined in Section 3. The need for a transport-level coordination mechanism is motivated by the needs of C-to-C applications like tele-immersion distributed sensor arrays, and CSCW, which represent an important class of future Internet applications. The specific design of CP is our attempt to satisfy those needs.

5.2 Design Justification

We anticipate several questions about why we designed CP in the manner that we have and justify its design below.

5.3 *Why do this below the transport-level?*

The primary reason for implementing CP below the transport-level is to preserve end-to-end semantics of the transport-level. Another possible approach would be to deploy CP at the application level by having all streams of a cluster sent to a multiplexing agent which then incorporated the data into a single stream sent to a demultiplexing agent on the remote cluster. This approach, however, has several drawbacks. By breaking the communication path into three stages, end-to-end semantics of the individual transport-level protocols have been severed. This approach also mandates that application-level control is centralized and integrated into the multiplexing agent.

A secondary reason for implementing CP below the transport-level is because that is where the CP mechanisms logically belong. The transport-level is associated with the end-to-end semantics of individual streams. The network-level protocol (i.e., IP) is associated with the next-hop forwarding path of individual packets. CP deals with streams

that are associated together and share a significant number of hops along their forwarding paths, but do not share exact end-to-end paths. This relaxed notion of a stream bundle logically falls between the strict end-to-end notion of the transport-level and the independent packet notion of the network-level.

5.4 *Why do this at packet granularity?*

By using every packet from every flow of data associated with a C-to-C application we can achieve fine-grained sampling of current network conditions. Fine-grained sampling is important because Internet network conditions are highly dynamic at almost all time scales. Sampling across the flows as an aggregate allows each transport-level protocol to have a more complete picture of current network conditions. Also, since each packet payload needs to be delivered to its intended endpoint destination, each packet provides a convenient and existing vehicle for communicating network condition information to the transport-level protocols and application processes that will react to changing conditions.

5.5 *Why not have the endpoints exchange information?*

Another possible architecture for a transport-level protocol coordination mechanism would be to exchange delay, loss, and bandwidth information among endpoints. This presupposes, however, that the endpoints are aware of each other. Our CP design, on the other hand, allows for loosely-coupled C-to-C applications in which this may not be the case. Endpoints of a C-to-C application are only required to use the same cluster identifier which can be distributed through a naming service like DNS or some other application-specific mechanism. The aggregation points are a natural place for cluster-wide accounting of loss, delay, and bandwidth since all packets of the cluster will pass through them. The design of CP is cognizant of the additional processing load this places on the aggregation point which is reflected in its relatively simple accounting mechanism.

5.6 Related Work

The ideas behind CP were primarily inspired by the Congestion Manager (CM) architecture developed by Balakrishnan [2]. CM provides a framework for different protocols to share information concerning network conditions. The CM is an excellent example of a coordination mechanism, but operates only when the transport-level flows share the entire end-to-end path.

In [6], Kung and Wang propose a scheme for aggregating traffic between two points within the backbone network and applying the TCP congestion control algorithm to the whole bundle. The mechanism is transparent to applications and does not provide a way for a particular application to make interstream tradeoffs.

Pradhan et al. propose a way of aggregating TCP connections sharing the same traversal path in order to share congestion control information [8]. Their scheme takes a TCP connection and divides it into two separate ("implicit") TCP connections: a "local subconnection" and a "remote subconnection." This scheme, however, breaks the end-to-end semantics of the transport protocol (i.e., TCP).

Active networking, first proposed by [11] allows custom programs to be installed within the network. Since their original conception, a variety of active networking systems have been built [14, 1, 3, 12, 15, 5, 7, 10]. They are often thought of as a way to implement new and customized network services. In fact, CP could be implemented within an active networking framework. Active networking, however, mandates changes to routers along the entire network path. This severely hinders deployment. CP requires changes only at the endpoints and at the aggregation points.

6 Summary

In this position paper, we motivated the need for transport-level protocol coordination for a particular class of distributed applications that we described as cluster-to-cluster applications. In particular, C-to-C applications are characterized by many independent, but semantically related, flows of data between clusters of computation and communication devices. This application architecture requires that end-to-end transport-level semantics be preserved, while at the same time, providing each stream with a consistent view of current networking conditions. To achieve this, each transport-level protocol needs measures of congestion, delay, and loss for the aggregate bundle of data flows.

We described the design of the Coordination Protocol (CP) which provides for the unique networking requirements of C-to-C applications. The main features of CP are:

- Transport-level protocol independent.
- Locally deployable.
- Fine grained sampling of network conditions.
- Flexible.
- Supports dynamic membership and configuration.

CP provides cluster endpoints with a consistent view of network conditions, as well as cluster membership and bandwidth usage information. With this information, a C-to-C application can make coordination decisions according to specific objectives and a privileged understanding of application state. CP will facilitate the development of the next generation of Internet applications.

References

[1] D.S. Alexander et al. Active bridging. *Proceedings of SIGCOMM'97*, pages 101–111, September 1997.

[2] Hari Balakrishnan, Hariharan S. Rahul, and Srinivasan Seshan. An integrated congestion management architecture for internet hosts. *Proceeding of ACM SIGCOMM*, September 1999.

[3] D. Decasper et al. Router plugins: A software architecture for next generation routers. *Proceedings of SIGCOMM'98*, pages 229–240, September 1998.

[4] J. Grudin. Computer-supported cooperative work: its history and participation. *Computer*, 27(4):19–26, 1994.

[5] M. Hicks et al. Plannet: An active internetwork. *Proceedings of INFOCOM'99*, pages 1124–1133, March 1999.

[6] H.T. Kung and S.Y. Wang. Tcp trunking: Design, implementation and performance. *Proc. of ICNP '99*, November 1999.

[7] E. Nygren et al. Pan: A high-performance active network node supporting multiple code systems. *Proceedings of OPENARCH'99*, 1999.

[8] P. Pradhan, T. Chiueh, and A. Neogi. Aggregate tcp congestion control using multiple network probing. *Proc. of IEEE ICDCS 2000*, 2000.

[9] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. *Proceedings of ACM SIGRAPH 98*, 1998.

[10] B. Schwarts et al. Smart packets for active networks. *Proceedings of OPENARCH'99*, 1999.

[11] D. L. Tennenhouse and D. Wetherall. Towards an active network architecture. *Multimedia Computing and Networking*, January 1996.

[12] J. van der Merwe et al. The tempest - a practical framework for network programmability. *IEEE Network Magazine*, 12(3), May/June 1998.

[13] M. Weiser. Some computer science problems in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, July 1993.

[14] David Wetherall. Active network vision and reality: lessons from a capsule-based system. *Operating Systems Review*, 34(5):64–79, December 1999.

[15] Y. Yemini and S. da Silva. Towards programmable networks. *International Workshop on Distributed Systems Operations and Management*, October 1996.

[16] T.-P. Yu, D. Wu, K. Mayer-Patel, and L.A. Rowe. dc: A live webcast control system. *To appear in Proc. of SPIE Multimedia Computing and Networking*, 2001.