Proceedings of the 5th IEEE International Conference on
High-Speed Networks and Multimedia Communications HSNMC'02, July 2002, Jeju, Korea.

Flow Assignment in A Self-Organizing Video Stream that Auto Morphs Itself while in Transit via a Quasi-Active Network

Javed I. Khan, Patrick Mail and Seung S. Yang

Media Communications and Networking Research Laboratory
Department of Computer Science
Kent State University, 233 MSB, Kent, OH 44242
javed|pmail|ssyang@kent.edu

Abstract

In this paper we present an algorithm for flow assignment in active nomadic computation. We present the algorithm with a novel active application – a cognizant video streaming mechanism which is capable of negotiating local network state based rate and let the video propagate over extreme network with highly asymmetric link and node capacities. As a platform the stream uses the computing power of a quasi-active network. As a result the passing video stream appears as a self-organizing stream, which automatically senses the network asymmetry and adapts itself as the packets diffuse via the active subnet.

Key Words: Diffusion Computing, Transcoding, Adaptive Video Streaming, Asymmetric Internet.

1. Introduction

Adaptation is a fundamental phenomenon in natural systems. The engineering of any large and complex system intrinsically requires inbuilt ability of its components to adapt. Internet has already grown into a meganet with global reach. With its growth the capacity differential between various parts of the networks and the capabilities of egress devices has increased as well [1,10,11]. Historically the Internet architecture has been conceived to cope with the heterogeneity of network standards. It now appears that a second era is evolving. Next generation of Internet will have to deal with more intrinsic heterogeneity— the asymmetry of hard resources such as bandwidth, switching, and egress device capability. Asymmetry can evolve from the fundamental physical limitations such as the power crunch in an intergalactic network element, or from something as close and insurmountable as socio-economical disparity- the digital divide. With the expansion and aging the relative size of the pool of older yet functional devices will also increase. The resource asymmetry can take the form of both bandwidth and computing capacity differential.

Without efficient adaptation ability systems tends to loose ubiquity in a large asymmetric environment [13,14]. It is important to build protocols, applications and services which will operate seamlessly across networks and devices of widely varying capabilities and will not cease to operate at the splice points.

It is important to note growing asymmetry is not only a concern for egress-applications. It will increasingly become a concern in network protocols design as well. In the asymmetric situation the capability to execute a specific transport, routing, error correction algorithm may be constrained not only by the switch processor speed, but other factors such as power limitation (mobile devices, sensor network), or availability (periodic occlusion in line-of-sight communication, or in vehicular network). Notably, concern about computational resource capability at intermediate nodes is increasing in all layers of networking. Power limited devices are requiring the design of power efficient low level protocols. In the upper layers, sophisticated computations are now being added in the data path of information flow in the form of various custom and adaptive information processing services [8]. Recently, proposed techniques such as programmable network or active networking [4,6,12] are investigating how protocol elements can be customized to meet the varying needs. In this research we are particularly investigating how a communication super-structure can be maintained over an asymmetric infrastructure, where protocols and applications can be built and deployed without any hard assumption about the computational power available in the nodes. In this paper we present an algorithm for flow computation assignment on an asymmetric network with heterogeneous link and node capacities. We have recently implemented an auto-morphing MPEG-2 ISO-13818-2 [3] symbiotic video streaming system with automatic computation migration ability. We present the algorithm in the context of this adaptive video streaming system.

1.1 Adaptive Streaming

We call the stream the **Self-Organizing Object-Based Network Embedded Transcoding** (SONET) stream. It is a prototype for an information flow network, which can serve video from a source to clients with heterogeneous capabilities via network with unequal capabilities. Fig-1 illustrates the

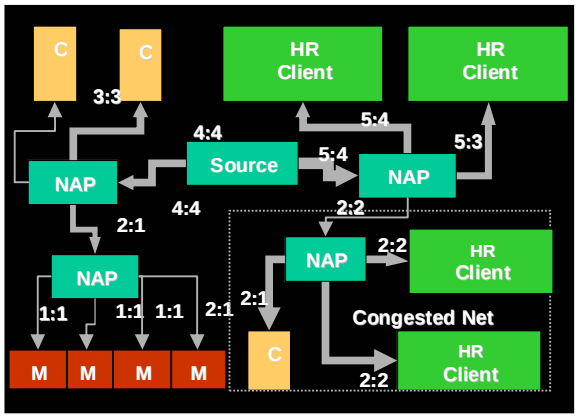

Fig-1 Video distribution in a large network with heterogeneous links and clients. Labels show the capacity and flow. NAPs are the *constriction points.*

situation. It has regular (C), mobile (M) and high resolution (HR) clients. The link labels identify the varying link capacities and flow. Well placed network adaptation units (NAPS) can enable optimum data flow satisfying all clients at their desired rate. SONET represents an adaptive system with substantial software and systems engineering complexity, and involves high volume data communication with temporal quality constraints, reactivity with network state, and infusion of sophisticated domain specific processing. The general information flow distribution model of SONET is valid not only for live video multicast, but for any information distribution scheme from single source to multiple clients, whether simulcast or not. For example, for stored video distribution the NAP can be augmented with hybrid cache [7].

The internal transcoding mechanism [5] of SONET stream has been formulated with an information flow centric view rather than component centric view. The focus is on the flow of video information. The associated transformations (and the associated transcoding operations) take place in a distributed way on its flow path. The key to this design is the modular decomposition of the required transcoding operation—where instead of a single monolithic implementation we took an experimental approach of building them as dynamic hyper-linkable capsules with easily separable data flow optimized concurrent modules, and these active modules themselves can 'flow' with the video flow. The stream uses a quasi-active subnet for computation. In this model (Fig-2) a sparse set of active stations in a sea of regular router cooperatively computes the transformations. As a result the SONET stream appears as a self-organizing stream, which automatically senses the network asymmetry and adapts itself as the packets diffuse via the active subnet.

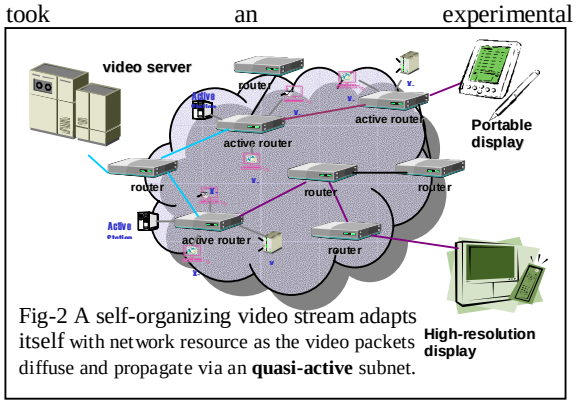

Fig-2 A self-organizing video stream adapts itself with network resource as the video packets diffuse and propagate via an **quasi-active** subnet.

1.2 Adaptation in SONET

As an experimental adaptive system in SONET we have carefully selected set of adaptive attributes those are representatives of major class of adaptive behaviors. In the top level it shows (a) adaptation with respect to bandwidth asymmetry and (b) adaptation with respect to node capacity asymmetry. In each class, it has further selected sub-techniques. The transcoding operation itself acts as a means of *rate adaptation.* For the *compute power adaptation* SONET uses two mechanics-- (ii) **modular self-organization** and (ii) **computation diffusion**. These two techniques represent different levels of reactivity with the network. To adapt to the available computation resource the first uses domain specific technique to cutback on the internal computation. However, when the capacity of a single node is insufficient to support the lowest transformation the second mechanism enables SONET to sweep computation power from other nodes and diffuse the computation onto multiple nodes in the network neighborhood. These two mechanisms help SONET streams to be uniquely ubiquitous, and to operate on networks with wide variety of computational and bandwidth constraints. A key technique behind the nomadicity and self-organization of the stream is a novel active component mapping scheme that dynamically match the topology's computation and communication capacity with the requirements of the adaptive stream. In this paper, we focus on the SONET mechanism for flow assignment on the active net based on these joint (link and node) resource constraints.

1.3 Related Work

There is little previous work that considers joint dynamic mapping of network flow. The classical algorithms for flow mapping are mostly based on maximum flow/shortest path evaluation [2]. However, these flow algorithms can not be directly applied due to the side extension possibility. Also, the classical flow algorithms may not be easily amenable to the hierarchical organization of internetwork graphs. Researchers in active network have investigated several mechanisms for capsule distribution over a flow path [6,7,15]. The variable nature of capsule's computing cycle requirement is uniquely important for active network. However, no previous work has been reported that considers the capsules/network resource mapping. The closest set of work which has considered some form of optimized distribution of network embedded task can be found in Web caching research, which considered storage and delay optimization [7].

The proposed mapping mechanism has two novel characteristics. Unlike other flow mapping algorithms,

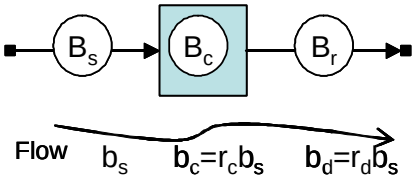

Fig-3 Capacity and Flow metrics of a node.

we address joint bandwidth and node computational capacity constraints. Secondly, we propose flow mapping technique that can accommodate the hierarchical, irregular, and dynamic nature of the internet. Indeed we will present a hierarchical network abstraction principle, which can be used as a building block towards designing a family of joint constraints satisfying flow mapping algorithms.

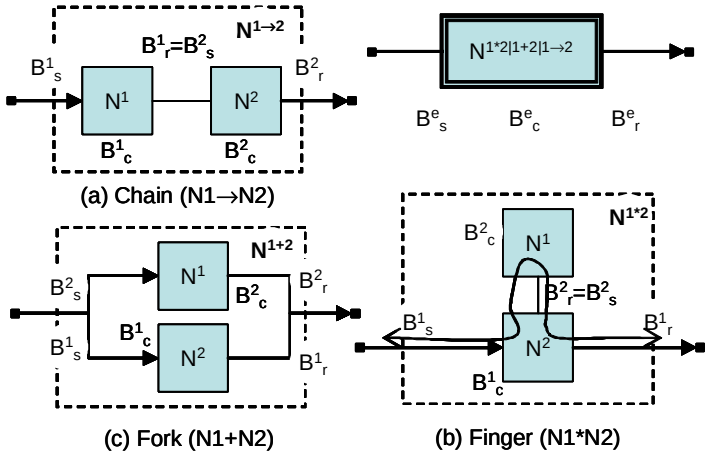

Fig-4 Chain, Fork and Finger Primes. A ny network graph can be recursively factored into smaller graphs allowing recursive assignment of partial load.

2. Mapping of Active Flow

The SONET component placement mechanism has two stages. The first is the determination of the *constriction points*—or the approximate locality where the rate transcoding has to be performed. A distribution scheme can have multiple constriction points. The second stage is the casting of the transcoder elements in the neighborhood of the constriction point. In this paper we address this second stage. The first stage issues are domain specific [17]. The placement logic requires the approximate network topology and the quality-of-network descriptions of the involved links and nodes. The approximate topology search algorithm searched for the distribution tree between the designated source and sinks. Once, the constriction points are determined the component deployment process requires finer network map nearby. At this stage the algorithm runs a k-best path search between the source and constriction-point. Finally it runs a d-radius neighborhood search to include few additional nodes into the picture. It then accepts the *component connection map* [4]. Once the session resource is requested the mapping, algorithms determines the best mapping for the components. Below we now describe the flow mapping model.

2.1 Model

Each mapping involves a session, the active platform and an active transport channel.

A *session* is modeled with a source flow *F* bytes/second. F denotes the data volume at the source.

The *platform network* is characterized by a graph N(V,L,M), where v^i is the active node, l^{ij} are the overlay links and M is the capacity metric. In M each link has a bandwidth attribute B^{ij} bytes/sec and each node has a compute power attribute B^i flops.

The flow processing channel is characterized by a *process dependency graph* with process stage nodes $P=\{p^j\}$. The application does not require the advance knowledge of the actual session flow volume or the capacity of the platform network. Each of the process stages p^j is modeled using its computational and outflow requirements per unit of inflow respectively denoted by r^j_c flops/bps and r^j_d. If b_s bps is the source flow, then $r^j_c.b_s$ flops is the required computational power and $r^j_d.b_s$ bps is the outflow. The model is illustrated in Fig-3.

Given the above model, the assignment problem is defined as following-- given a session A, an active application P, and a network *N(V,L,M)* determine the assignment of p^j on nodes v^i.

We have developed a novel technique of the network factorization to assign the flow processing task on the network. Any given network is factorized into a set of three sub-networks: (i) fork net (+), (ii) chain net (→), and (iii) finger net (*). Fig-3 shows the sub-networks. Any network graph can be factored in terms of these prime set. We demonstrate three abstraction rules for capacity aggregation for these three abstract primes. The flow assignment algorithm uses recursive hierarchical factorization to determine the assignment.

2.2 Abstraction Grammar

2.2.1 Chain Abstraction:

Let N^1 and N^2 are two sequential subnets in a pathway from source to destination. Let $\{B^1_s, B^1_c, B^1_d\}$ and $\{B^2_s, B^2_c, B^2_d\}$ are respectively the aggregate inflow, computation, and outflow capacities of the two nets. Let, B^{2e}_c is the abstraction equivalent computing capacity of the second net (Fig-3(a)). If $B^{1\to2}$ is the abstraction of these two nets, then the aggregate abstract capacities are given by $\{B^{1\to2}_s, B^{1\to2}_c, B^{1\to2}_d\}$, where:

$$B_c^{2e} = \min\{r_c.B_s^2, B_c^2, \frac{r_c}{r_d}B_d^2\}$$

$$B_c^{1\to 2} = B_c^1 + B_c^{2e} \quad \ldots(1a)$$

$$B_s^{1\to 2} = B_s^1 \quad \ldots(1b)$$

$$B_d^{1\to 2} = B_d^2 \quad \ldots.(1c)$$

2.2.2 Finger (Stubnet) Abstraction

Let N^1 is subnet in the pathway from source to destination, and N^2 is a finger subnet which is reachable to the flow pathway via N^1. Let $\{B^1_s, B^1_c, B^1_d\}$ and $\{B^2_s, B^2_c, B^2_d\}$ are respectively the abstract inflow, computation, and outflow capacities of the two networks (Fig-3(b)). Let, B^{2e}_c is the abstraction equivalent computing capacity of the finger subnet. If B^{1*2} is the abstraction of these two networks, then its aggregate abstract capacities are given by $\{B^{1*2}_s, B^{1*2}_c, B^{1*2}_d\}$, where:

$$B_c^{2e} = \min\{B_c^2, \frac{r_c}{1+r_d}B_d^2\}$$

$$B_c^{1*2} = B_c^1 + B_c^{2e} \quad \ldots(2a)$$

$$B_s^{1*2} = B_s^1 \quad \ldots(2b)$$

$$B_d^{1*2} = B_d^2 \quad \ldots.(2c)$$

2.2.3 Fork Abstraction

Let N^1 and N^2 are two concurrent subnets in a pathway from source to destination. Let $\{B^1_s, B^1_c, B^1_d\}$ and $\{B^2_s, B^2_c, B^2_d\}$ are respectively the abstract inflow, computation, and outflow capacities of the two nodes. Let, B^{1e}_c, B^{2e}_c is the abstraction equivalent computing capacity of the second subnet (Fig-3(c)). If B^{1+2} is their abstraction, then its aggregate abstract capacities are given by $\{B^{1+2}_s, B^{1+2}_c, B^{1+2}_d\}$, where:

$$B_c^{1e} = \min\{r_c.B_s^1, B_c^1, \frac{r_c}{r_d}B_d^1\}$$

$$B_c^{2e} = \min\{r_c.B_s^2, B_c^2, \frac{r_c}{r_d}B_d^2\} \quad \ldots(3a)$$

$$B_c^{1+2} = B_c^{1e} + B_c^{2e}$$

$$B_s^{1e} = \min\{B_s^1 - \frac{1}{r_c}B_c^{1e}, B_d^1 - \frac{r_d}{r_c}B_c^{1e}\}$$

$$B_s^{2e} = \min\{B_s^2 - \frac{1}{r_c}B_c^{2e}, B_d^2 - \frac{r_d}{r_c}B_c^{2e}\}$$

$$B_s^{1+2} = B_s^{1e} + B_s^{2e} + \frac{1}{r_c}B_c^{1+2} \quad \ldots(3b)$$

$$B_d^{1+2} = B_s^{1e} + B_s^{2e} + \frac{r_d}{r_c}B_c^{1+2} \quad \ldots.(3c)$$

2.3 Assignment Constraints

Let, $A_c = r_c.I_s$ is the flow to be assigned. Let $\{B_s, B_c, B_d\}$ are respectively the abstract inflow, computation, and outflow capacities of a unit network $N(V, L, M)$. Then the maximum assignable flow process share of p^i in this network can is given by:

$$B_c^e = \min\{r_c.B_s, B_c, \frac{r_c}{r_d}B_d\} \quad \ldots(4b)$$

Proof: Essentially, there can be three constraining situations. This network can be (i) inflow-capacity bound, (ii) compute capacity bound, or (iii) outflow capacity bound. The maximum possible assignment is the minimum assignment possible in either of these situations. Correspondingly, for this assignment consumed inflow and outflow capacities will be:

$$B_s^e = \frac{1}{r_c}B_c^e \quad \ldots(4b)$$

$$B_d^e = \frac{r_d B_c^e}{r_c} \quad \ldots(4c)$$

2.4 Reminder Abstract Network

Correspondingly, after the assignment the reminder network $N'(V', L', M')$ will have the capacities:

$$B_c^{'} = B_c - B_c^e \quad \ldots(5a)$$

$$B_s' = B_s - B_s^e \quad \ldots(5b)$$

$$B_d^{'} = B_d - B_d^e \quad \ldots.(5c)$$

After the assignment the reminder flow will be:

$$A_c' = A_C - B_c^e, r_c, r_d \quad \ldots.(5d)$$

After the assignment one of these four quantities must be zero. If the network is compute-bound, the unit network with zero capacity can be removed and the inflow and outflow links l_s and l_d can be substituted by a

Network Factorization/ Capacity Abstraction Algorithm

Take a Network N(V,L,M) {
1. Find the k-shortest path from the source to sinks.
2. If no path terminate "DONE with failure".
3. For each j-th path:
   a. For each node i v(j,i) in the j-th chain.
      i. Apply Finger Abstraction of the node: compute F(i,j).
   }
}
4. Now the graph has k simple (no finger) chains with F nodes.
5. For each chain in F
   a. Begin from the last node nth of the chain.
      i. Apply Chain Abstraction nth→(n-1)th nodes.
   }
}
6. Now the graph has m singleton (one node) Chains.
7. Now Assign 'chain share' of the flow along the best path.
8. For each node in the chain of the selected path:
   a. Assign 'finger share' of the flow.
9. Obtain reminder flow A'.
10. If no reminder task terminate "DONE with success".
11. Else, {
   a. Obtain reminder graph N'(V',L',M').
   b. continue from step-1 on N' with flow A'.

}

pass through link l_p $N'(V', L', M')$ with capacity:

$$B'_p = \min(B'_s, B'_d) \quad \text{....(5e)}$$

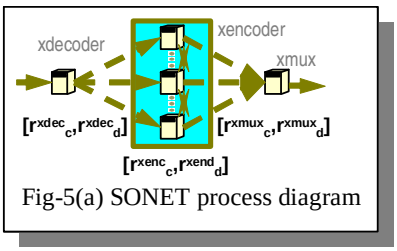

Fig-5(a) SONET process diagram

2.5 Network Factorization Algorithm

The advantage of our factorization/abstraction based assignment approach is two fold. First it allows hierarchical abstraction of network capacity and is suitable for typical internet scenario. Secondly, it can be applied as a building block for many different assignment strategies. Algorithms can be designed to suit the requirements of a particular application. Following is a choice of algorithm for SONET. Here the domain constraint specifies that the transcoding task (rate reduction) must be performed prior to the *constriction point* (the bottleneck link). Thus, the driver algorithm begins from the destination node and backfills the computation.

The complexity of the algorithm is O(V+E). In each abstraction step the algorithm eliminates either one link or a node.

3. SONET System

3.1 Component Architecture

The SONET channel components are (a) GOP-Encoder (GOP-ENC), (b) decode- demultiplexer (DE-DEMUX) and (c) GOP multiplexer (GOP-MUX). The GOP-ENC is the XCODER. One GOP-DEMUX and one GOP-MUX are activated at the designated sub-net entry and exit points. GOP-ENC is the most computing intensive unit and requires diffusion computing. Fig-5(a) shows a sample SONET process dependency graph.

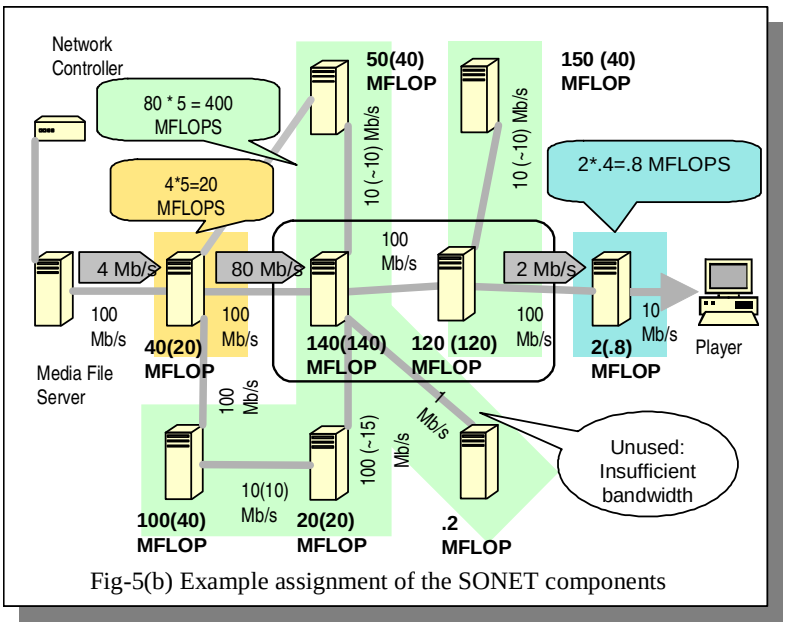

Fig-5(b) Example assignment of the SONET components

3.2 Dynamic Unit Deployment

For dynamic reorganization, multiple instances of the GOP-ENC units are kept in a dormant state in designated active nodes. Active modules are then connected by a feedback system. The GOP-MUX module senses the resultant frame rate. After the initial deployment each unit sends back the computing and bandwidth information from the node's immediate neighborhood by dynamic estimation. The mapping algorithm then continually determines the assignment. If the aggregate capacity falls below a given target frame rate, it then sends signals to DE-DEMUX unit, which then activates dormant GOP-ENC units in the active subnet to join in.

3.3 Example Deployment

Fig-5(b) shows a sample quasi-active network between two constriction points and SONET component mapping for this network for a F=4 Mbps source session with 50% rate reduction. We placed an MPEG-2 server in high-speed segment and a simple MPEG-2 client into the wireless network. The network in-between contained a small quasi-active subnet. SONET connects the server and the client.

The example assumes that DE-DEMUX, GOP-END and GOP-MUX units to have capacity metrics {5 FLOPS/bps, 20}, {5 FLOPS/bps:1/40}, and {.2 FLOPS/bps:1}. The consumed capacities are shown in bracket. For the given properties, finger nodes attached with 10mbps links can accommodate about

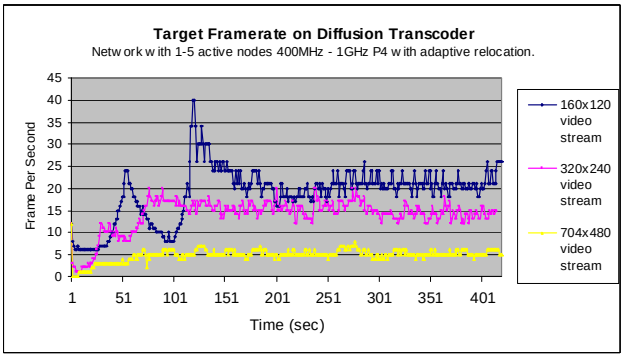

Fig-7 The adaptive growth of frame rate based on the GOP arrival times recorded at the GOP-MUX unit. The adaptive behavior is noticeable at the step like increments at the very beginning.

40 FLOPS of task.

3.4 Performance

Here we share some experiment results of this system from some sample run. This performance is given for its user space execution and it does not use any domain specific optimization (such as motion vector bypass).

Fig-7 shows the frame-rate observed in their sample run on a small uncontrolled (with background computational and communication load) active network consisting of 5 active routers (with capacity ranging from 400 MhZ ~ 1.5 GhZ P4 processors, and the interconnects were 10/100 Ethernets with uncontrolled cross traffic). We let the system auto deploy itself and find optimum mapping. Figure-7 plots the frame/ second statistics recorded at the GOP-MUX unit. It plots the performance for three streams with 16x120, 320x240 and 704x480 frame sizes. The computation load heavily depends on the number of macro-blocks or frame size. A dynamic estimation algorithm was used to determine the computing/byte ratios. Based on the frame size the frame transcoding rate varied from 30-5 frames/second. The adaptive behavior is noticeable at the step like increments at the very beginning. Initially the channel used only one active node. As the single node was unable to sustain the target rate, it auto-deployed additional active nodes. For example for 704x480 video the second and the third nodes were deployed some time before 20th and 60th seconds respectively. These delays represent the full feedback and effectuation delays. It includes (i) the time to detect insufficiency, (ii) the time for stream auto deployment, and (iii) the time it takes the new results to appear at the MUX. As evident from the jumps only three active paths were available. This performance of user space realization of unoptimized SONET should provide some valuable insight about the computational capabilities that might be sustainable on an active channel system.

4. Discussions

SONET streaming demonstrates the potential power of a new genre of active applications with intelligent adaptation ability. Within the scope of this paper we could only address the issues pertaining to the dynamic component mapping. The mapping technique can be potentially used in a wide variety of emerging networked scenarios. However, nomadic flow management is a complex problem. Rate adaptation in any perceptual information flow is a three way tradeoff between compute power, bandwidth reduction ratio, and quality of video. We are currently conducting additional experiments to characterize such trade-off characteristics of SONET. There is also relationship between jitter sensitivity and task assignment. We have recently developed dynamic jitter reduction algorithms for multi-path active computation. Additional references on these issues can be found in [9,16,17].

The work is currently being funded by the DARPA Research Grant F30602-99-1-0515 under its Active Network initiative.

5. References

[1] Amir, Elan, Steven McCanne, and Randy Katz, Receiver-driven Bandwidth Adaptation for Light-weight Sessions, Proceedings of ACM Multimedia '97, Seattle, WA, Nov 1997.

[2] T. Corman, Leiserson, Rivest, and Stein, Algorithms, 2nd Edition, McGraw Hill, 2001.

[3] Information Technology- Generic Coding of Moving Pictures and Associated Audio Information: Video,ISO/IEC International Standard 13818-2, June 1996

[4] Javed I. Khan, S. S. Yang, Medianet Active Switch Architecture, Technical Report: 2000-01-02, Kent State University, [URL http://medianet.kent. edu/ technicalreports.html, also mirrored at http:// bristi.facnet.mcs.kent.edu/ medianet]

[5] Keesman, Gertjan; Hellinghuizen, Robert; Hoeksema, Fokke; Heideman, Geert, Transcoding of MPEG bitstreams Signal Processing: Image Communication, Volume: 8, Issue: 6, pp. 481-500, September 1996,

[6] Tennenhouse, D. L., J. Smith, D. Sincoskie, D. Wetherall & G. Minden., "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, Jan 97, pp 80-86

[7] M. Rabinovich & O. Spatscheck, Web Caching and Replication, Addison Wesley, NY, 2002.

[8] Wei-Ying Ma, Bo Shen and Jack Brassil, "Content Services Network: The Architecture and Protocols", Int. workshop on web caching and content distribution, June 2001.

[9] Javed I. Khan, & S. S. Yang, Made-To-Order Custom Channels for Netcentric Applications over Active Network, Proc. Of the Internationaal Conf. On Internet and Multimedia Systems and Applications, IMSA 2000, Nov 2000, Las Vegas, pp22-26.

[10] Gary Stix, The Triumph of the Light, Scientific American, January 2001, pp31-35 [HTTP:// http://www.sciam.com/ 2001/ 0101issue/ 0101stix.html]

[11] Peter Forman and Robert W. Saint, Creating Convergence, Scientific American, November 2000,pp.10-15[HTTP:// http://www.sciam. com/ 2001/0101issue/0101stix.html]

[12] Xuebin Xu, "A Survey of Ideas in Programmable Networks", Technical Report: 2002-01-01, Kent State University, [available at URL http://medianet.kent.edu/ technicalreports. html, also mirrored at http:// bristi.facnet.mcs.kent.edu/ medianet]

[13] L. Klienrock, Nomadic Computing, Proc. IFIP/ICCC Intl.Conf. on Information Network and Data Communication, Chapman & Hall, London, 1996, pp.223-233.

[14] R. Want and B. Schilit, Expanding the Horizons of Location Aware Computing, IEEE Computers, August 2001, v.34,n.8, p.p31-35.

[15] Y. Yemini and S. da Silva. Towards Programmable Networks. In Intl. Work. on Dist. Systems

Operations and Management, Italy, Oct. 1996. [URL: http://www.cs.columbia.edu/dcc/netscript/Publications/publications.html, Last retrieved: 11/02/00]

[16] Javed I. Khan and Asrar Haque, An Active Programmable Communication Harness for Measurement of Composite Network States, IEEE International Conference on Networking, ICN' 2001, pp628-638.

[17] Javed I. Khan & S. S. Yang, Resource Adaptive Nomadic Transcoding on Active Network, International Conference of Applied Informatics, AI 2001, February 19-22, 2001, Insbruck, Austria, ISBN:0-88986-280-X (307), pp262-267.