A Markovian Approach to Multi-path Data Transfer in Overlay Networks

Vinh Bui, *Student Member, IEEE,* Weiping Zhu, *Member, IEEE,* Alessio Botta, *Student Member, IEEE,* and Antonio Pescapé, *Senior Member, IEEE*

Abstract—The use of multi-path routing in overlay networks is a promising solution to improve performance and availability of Internet applications, without the replacemenet of the existing TCP/IP infrastructure. In this paper, we propose an approach to distribute data over multiple overlay paths that is able to improve Quality of Service (QoS) metrics, such as the data transfer time, loss, and throughput. By using the Imbedded Markov Chain technique, we demonstrate that the system under analysis, observed at specific instants, possesses the Markov property. We therefore cast the data distribution problem into the Markov Decision Process (MDP) framework, and design a computationally efficient algorithm named Online Policy Iteration (OPI), to solve the optimization problem on the fly. The proposed approach is applied to the problem of multi-path data distribution in various wired/wireless network scenarios, with the objective of minimizing the data transfer time as well as the delay and losses. Through both intensive ns-2 simulations with data collected from real heterogeneous networks and experiments over real networks, we show the superior performance of the proposed traffic control mechanism in comparison with two classical schemes, that are Weighted Round Robin and Join the Shortest Queue.

Index Terms—Multi-path Data Transfer, Markov Decision Processes, Overlay Networks, Heterogeneous Networks.

1 Introduction

Despite the rapid development in network infrastructures, applications requiring quality-assured data transfers on the best-effort Internet continue to suffer from limited bandwidth, highly varying delay and losses. In literature several studies suggested that an application can increase its aggregate throughput, and reduce its end-to-end delay and losses, by distributing data over multiple paths [1], [2], [14], [16], [20], [36]. Nonetheless, the effort to support multiple paths at the IP or lower-level layers, e.g. in [13], [40], has not been attractive so far, since the deployment of a new IP-incompatible network infrastructure is required. Recently, application-level overlay networks have been used to establish multi-path capable environments over the Internet [4], [5], [31]. The overlay network approach avoids modifying the existing network infrastructure [13], [40]. However, since an overlay network is deployed on top of an underlay network, the overlay traffic has to compete with the underlay one for resources, and then congestions can occur. As a result, a significant challenge in providing high performance data transfers - by using multi-path overlay networks - is to infer the congestion states of overlay paths and to deal with fluctuations in overlay path performance. This paper addresses such challenge using a path state monitoring mechanism that complies with the end-to-end principle and captures congestion states of overlay paths. Based on the captured path states, a traffic control mechanism based on Markov Decision Processes (MDPs) is used to simultaneously route traffic over multiple overlay paths, while optimizing some QoS metrics such as end-to-end delay, jitter and consecutive losses. This control mechanism has the ability to quickly adapt its transmission strategy to the dynamics of the underlying network. Consequently, it can avoid congestions by dynamically shifting traffic over multiple overlay paths. Such congestion avoidance ability comes naturally as a result of the adaptation or "learning" process. Hence, no explicit congestion avoidance mechanism is required. The proposed mechanism is designed to work above the transport layer where the decision timing is less critical and it can be used in situations like multimedia streaming applications and data distribution scenarios.

1.1 Related Work

In [18] the authors overview the main motivations, challenges, and techniques for *multi-path routing* at different levels (packet, flow) and in different operating scenarios (intradomain, interdomain) whereas the work presented in [37] is concerned with the experimental analysis of the possible *path diversity* (i.e. the number of routes available to transmit a packet between two hosts in a network) in ISP networks. *Optimal multi-path* data transfers on overlay networks have been studied in [14], [16], [36], [42]. In [36], Tao *et al.* demonstrated the potential of multiple overlay paths for improving Internet performance. They developed a path switching mechanism, that subsequently selects a potentially less congested path among several available paths to transmit data. The path selection is based on path congestion states,

- *Vinh Bui and Weiping Zhu are with the University of New South Wales, Australia. E-mail: {v.bui, w.zhu}@adfa.edu.au.*
- *Alessio Botta and Antonio Pescapé are with the University of Napoli Federico II, Italy. E-mail: {a.botta, pescape}@unina.it.*

Preliminary results within the same framework of this work have been recently published in [11], [8], [10].

predicted by a Markov model, that is trained with data collected through active measurements. A drawback of this approach is the overhead caused by active measurements, and the accuracy of the path model. Moreover, in many cases, sending data always to the less congested path is not the optimal policy . In [42], a multi-path controller is proposed that randomly chooses a transmission path following a throughput-proportional selection scheme. This path selection scheme is a randomized version of the well-known Weighted Round Robin (WRR) scheme [27] where the weights, in this case, are the path throughputs. A key issue of this scheme is how to accurately obtain the path throughput, by means of on-line measurements. However, we later show that, even with the correctly estimated path throughput, WRR is unlikely to be an optimal scheme for QoS metrics, such as loss, delay and jitter. At *transport layer*, the work [17] proposes an enhanced TCP scheme which works just above the transport layer opening different TCP connections towards the same host, whereas modifications to STCP aimed at supporting the effective use of multiple independent paths are proposed in [19]. In *wireless scenarios*, the work [12] presents an algorithm, borrowed from wireless scheduling problems, to distribute packets on different available paths while minimizing delay, maximizing throughput, and respecting the split proportions assigned by the routing algorithm. In [38] an analysis of path diversity at packet level in IEEE 802.11 scenarios with multiple-channel transmission is conducted, whereas the problem of multi-path routing in ad-hoc networks is analyzed in [26]. In the context of *data replication and collection*, the authors of [14] and [16] studied the problem of multi-path data transfer, where the objective was to minimize the *makespan*, i.e. the duration of the longest transfer. Both studies formulated the problem by using graph theory, where the graph is the overlay network topology, and the constraints are the path capacities. However, this approach has a number of potential issues, as admitted by the same authors [14]. First, the graph theory formulation assumed that the time required to transfer a unit of data through a link is fixed, which is unrealistic: in real networks, in the presence of background traffic, this transfer time is varying. Second, since end-to-end delay of a path is not always proportional to its capacity, using path capacities as constraints to minimize the *makespan* may not lead to an optimal solution. In such a dynamic environment, an approach based on a dynamic optimization framework like Markov Decision Processes (MDPs) is more appropriate. Markov Decision Processes [34] are a powerful mathematical framework for making decisions in a dynamic environment that exhibits Markov property[1]. In this framework, the control environment is modeled as a system of states. The system changes its state according to some state transition probability, which is a function of actions, i.e. decisions, taken by the decision maker. Outcome of each action is assessed by associating it with a reward. The goal of the decision maker is to maximize some function of rewards. Given an MDP, a decision is made by mapping an observed system state to an action according with a predefined policy. Once a decision policy is determined, actions are taken instantly without any extra computation. Therefore, MDPs can be used for time critical decision tasks, such as network traffic control.

1. Modern MDP-based decision frameworks, e.g. Reinforcement Leaning [35], can also deal with non-Markovian environments.

1.2 Our Contribution

In this paper we propose an approach to distribute packets over multiple overlay paths, which is based on Markov Decision Process, and allows to achieve better performance than Weighted Round Robin (WRR) and Join the Shortest Queue (JSQ). Thanks to the existence of Markov property in many network characteristics, e.g. arrival processes, loss, and delay [6], [9], [25], [33], MDPs have been used in the networking field to address various decision problems (e.g. [3], [11]). However, this does not mean that applying an MDP framework is straightforward. Major issues lie in the formulation of decision problems as an MDP and the solution of the MDP. In this work, we show its application to a set of data distribution problems, providing a solution to these issues. In particular, first, we design a simple network-path monitoring mechanism to track the state of the system under study. Second, by using the Imbedded Markov Chain technique, we demonstrate that the system, if observed at specific instants, possesses the Markov property. Third, using this result, we cast the data distribution problem into an MPD. Fourth, we propose a lightweight algorithm named Online Policy Iteration (OPI) to efficiently solve the MDP. Using OPI, we show the application of this novel distribution scheme in three network scenarios, where different objectives are pursued. We present the results of a complete simulation study, using ns-2[2], in both wired and wireless environments and with both UDP and TCP, which confirm superior performance of the approach against classical schemes, including WRR and JSQ. In addition, we also investigate issues related to the impact of the: (i) traffic source rate; (ii) number of nodes; (iii) number of paths; (iv) path diversity; (v) inaccurate estimation of the traffic parameters. Finally, we report preliminary results of an implementation over real networks indicating that our multi-path scheme is actually able to improve the performance of the communications in real scenarios.

2 Problem Definition

To contextualize the problem under study, we introduce the notion of Multi-path Routing Overlay Network (MPRON). An MPRON consists of Overlay Agents (also referred to as relays or proxies) deployed by one or several ISPs on top of the underlying IP routing network.

2. http://nsnam.isi.edu/nsnam/index.php

The aim of an MPRON is to provide multiple independent[3] overlay paths, connecting source and destination pairs of the underlying network. Hence, traffic can be routed between the network nodes through multiple paths, and the network resources can be more efficiently utilized. In the context of MPRONs, we want to solve the problem of optimal overlay data transfer in two scenarios. In the first scenario, an application (or user) wants to transfer an amount of data from a source to a destination. The application is fully aware of the existence of the multiple overlay paths and actively routes traffic over $M(M > 1)$ chosen paths. The data is segmented into constant-length[4] bins before being dumped with a constant rate λ into an application module called *traffic distributor*, which distributes the bins over M connections, i.e. sockets, representing the M overlay paths. In the second scenario, the application is not aware of (or has no capability to distribute traffic over) multiple paths. Subsequently, it simply routes traffic to an appropriate Overlay Agent, which then forwards it to the destination using multiple paths. In this scenario we assume that the data arrives at the Overlay Agent with an average rate of λ. Since the arrival process is discrete, it is approximated using a Poisson process [23] with inter-arrival times between consecutive bins following an exponentially independent and identically distributed (i.i.d.) random variable. It is worth underlining that the arrival process here is not purely Poisson, but similar to Markov Modulated Poisson: the arrival process is considered Poisson(λ) only in a period of time T, after which, λ may change to another value. The i.i.d. assumption is reasonable since Internet paths are stationary over a scale of minutes [43]. Without loss of generality, we assume that the amount of data to be transferred in both scenarios, in terms of number of bins, is a random variable ν governed by a Geometric distribution[5] with parameter γ:

$$P(\nu = n) = (1-\gamma)\gamma^{n-1}, (0 \leq \gamma \leq 1); \quad n = 1, 2, \ldots \quad (1)$$

Our objective is to construct a control strategy that specifies on which path, and at what time, each bin of data should be transferred to the destination to achieve the minimum transfer time for the whole data set.

3 System and States

Previous studies on similar problems ([12], [22], [30]) when applying decision frameworks, often assume perfect knowledge of the network path characteristics, e.g. loss, delay and throughput. To avoid this assumption, we design a path state monitoring mechanism, which reveals loss and delay conditions of the path by keeping track of data bins being transmitted. The proposed mechanism works as follows. Before the traffic distributor sends a bin over a selected path, the bin ID and a timestamp are recorded on a list of size K. This information is kept until the bin is correctly received. In this manner, given the data stream, the evolution of the number of bins on the list will implicitly reflect the path state in terms of Round Trip Time (RTT) and loss. In a real implementation, the information regarding the transmission success can be retrieved directly from the transport layer protocols, in case of connection oriented protocols, or be explicitly acknowledged, in case of connectionless protocols[6]. At a first glance, this can be seen as a considerable overhead. However we have verified that, even if this information is obtained every 10 to 15 bins, the results presented in Sec. 6 remain almost unchanged. Fig. 1 illustrates the path state monitoring mechanism. From the traffic distributor viewpoint it is easy to see

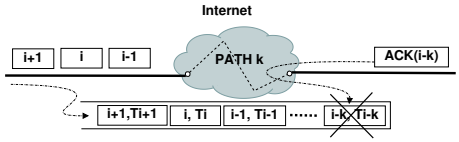

Fig. 1. Path state monitoring mechanism.

that, by using the path state monitoring mechanism, each path is modeled as a single server queue. The inter-departure intervals between "customers" in the queue are characterized by the distribution of RTT jitter[7]. Since each path is a single server queue, the system under study is a set of M parallel queues (thanks to the monitoring mechanism the buffer can be dynamically allocated in order to avoid overflows). The system states are created by vectors of "customers" queuing in the system. Fig. 2 depicts the system model. For each of the

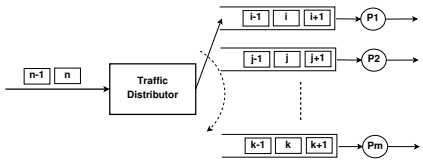

Fig. 2. The system model of M independent paths.

two scenarios presented in Sec. 2, we have a different arrival process: constant in the first, Poisson in the second. Having defined the system model and its states, in principle decisions can be made on how to optimally distribute data over the paths. Unfortunately, this system in general is not Markovian, i.e. the state of the system at an arbitrary time step may not depend exclusively on its previous state. Therefore, to apply the MDP framework, it is important that the Markov property of the system

3. In the context of this work, the term independent does not mean the paths are fully disjoint. Instead, they can share some non-congested links. Under what traffic condition the paths can be considered independent is discussed in Sec. 6.

4. The fixed size here is assumed to simplify the formulation of the problem, which can be easily generalized for variable size.

5. Vu *et al.* [39] supports this assumption in the case of overlay multimedia streaming.

6. To avoid the influence of congested reverse paths, acks should also carry a timestamp indicating the exact time in which the packet has been received.

7. RTT jitter is defined as the absolute difference between two consecutive RTT values i.e. $Jitter_i = |RTT_{i+1} - RTT_i|$.

is identified. In the following sub-section the Imbedded Markov Chain is applied to attain the Markov property of the system.

3.1 Imbedded Markov Chain

At the time instant when a data bin arrives at the traffic distributor, a decision has to be made on where to send the bin. Depending on the decision, the system may change its state. Therefore, the system behaviors observed at the arrival instant are important for the decision maker. To study these behaviors, the method of Imbedded Markov Chain [24] in combination with renewal theory [15] is applied. Let t_i be the arrival time of bin i at the traffic distributor. In accordance, let $Q_m(t_i)$ be the number of bins found in the m-th queue at t_i. Consequently, the system state at t_i is a vector $\{Q_m(t_i)\}, (m = 1 \dots M)$. We are interested in the evolution of $\{Q_m(t_i)\}, (i = 1, 2, 3 \dots \infty)$ under the impact of the data distributing actions taken by the decision maker at the traffic distributor. Assume that at arrival instant t_i, bin i is sent to the $m-$th queue by the decision maker. To eliminate the possible transitive period and simplify further discussions, it is assumed that a decision and its action can be completed instantly. As a consequence, the system will move immediately from state $\{Q_1(t_i), Q_2(t_i), \dots, Q_m(t_i), \dots, Q_M(t_i)\}$ to state $\{Q_1(t_i), Q_2(t_i), \dots, [Q_m(t_i) + 1], \dots, Q_M(t_i)\}$. From this moment until the arrival of the next bin, the evolution of $\{Q_m(t_i)\}$ depends only on the number of bins departing from each queue during the $[t_{i+1} - t_i]$ interval. In other words, if $P_{QQ'}$ denotes the probability that the system is in state $\{Q'_m(t_{i+1})\}$ at the arrival instant of bin $(i+1)$, then $P_{QQ'}$ equals to the probability that there are $Q_1(t_i) - Q'_1(t_{i+1})$ bins departing from the first queue, and $Q_2(t_i) - Q'_2(t_{i+1})$ bins departing from the second queue, and so on during the $[t_{i+1} - t_i]$ interval. Since these departure processes are independent[8], we can obtain $P_{QQ'}$ by taking the product of the probabilities. To compute the probability that there are $Q_m(t_i) - Q'_m(t_{i+1})$ bins departing from the m-th queue, some results from renewal theory are applied. Let τ_k and τ_{k+1} subsequently denote the departure times of bins k and $k+1$ from the m-th queue. Assume that the inter-departure intervals $[\tau_{k+1} - \tau_k]$ are i.i.d. random variables governed by a general distribution $g_m(x) = P(\tau_{k+1} - \tau_k \leq x)$. It is clear that the sequence of departure points $\{\tau_k\}$ forms an *ordinary* renewal process. Suppose the queuing process has attained a steady state. Subsequently, the sequence of departure points $\{\tau_k\}$ started from t_i forms the *equilibrium* renewal process of the above ordinary renewal process [15]. Fig. 3 illustrates the queuing process. According to renewal theory [15] as

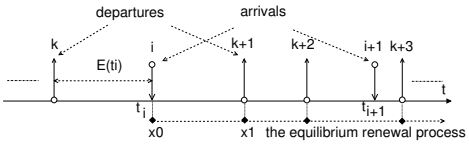

Fig. 3. The queuing process of the m-th queue.

applied for the *equilibrium* renewal process, the number of bins departing from the queue in the $[t_{i+1} - t_i]$ interval is proportional to the length of the interval and does not depend on t_i. Thus, $Q_m(t_{i+1})$ is fully determined by $Q_m(t_i)$ and does not depend on $E(t_i)$, which is the time elapsed since the last departure observed at t_i. Therefore, the random variable $Q_m(t_i)$ forms a discrete-state Markov chain embedded in the queuing process, whose state space is comprised of vectors of all of the possible numbers of bins in the queue. Hence, the Markov property of the system has been identified. Let β_j denote the probability of j bins, $j = 0 \dots K$, departing from the m-th queue in the $[t_{i+1} - t_i]$ interval. Since the sequence of departure times $\{\tau_k\}$ started from t_i forms an equilibrium renewal process, β_j is the probability of having j renewals during the $(0, t]$ interval, where $t = t_{i+1} - t_i$. In the first scenario, the bins arrive at a constant rate. Hence, $t = t_{i+1} - t_i$ is fixed and determined by the data rate λ. Subsequently, let $P[N(u) = r]$ denote the probability of having r renewals in an arbitrary interval $(0, u]$ of the equilibrium renewal process. We obtain:

$$\beta_j = P[N(t) = j] \tag{2}$$

In the second scenario, the arrival process is Poissonian with the average rate of λ. Hence, $t = t_{i+1} - t_i$ is a random variable governed by the exponential distribution with p.d.f. $\lambda e^{-\lambda t}$. Consequently, β_j is computed as follows:

$$\beta_j = \int_0^\infty P[N(t) = j] \lambda e^{-\lambda t} dt \tag{3}$$

Recall that the inter-departure intervals $[\tau_{k+1} - \tau_k]$ are i.i.d. random variables with a general p.d.f. $g_m(x)$. Let $G_m(x)$ be the corresponding cumulative distribution function of the inter-departure intervals and $G_m^{(n)}(x)$ be the n-fold convolution of $G_m(x)$. Let $\mu_m \equiv E[\tau_{k+1} - \tau_k]$ be the mean of the inter-departure time of the *m-th* queue. Subsequently, as pointed out in [15], the probability of having j renewals during interval $(0, t]$ of the equilibrium renewal process is obtained as follows:

$$P[N(t) = j] = \frac{1}{\mu} \int_0^t (P[N^o(u) = j - 1] - P[N^o(u) = j]) du$$
$$= \frac{1}{\mu} \int_0^t [(G_m^{(j-1)}(u) - G_m^{(j)}(u)) - (G_m^{(j)}(u) - G_m^{(j+1)}(u))] du$$

where $P[N^o(u) = r] = G_m^{(r)}(u) - G_m^{(r+1)}(u)$ is the probability of having r renewals during interval $(0, u)$ of the corresponding *ordinary* renewal process. Substituting received $P[N(t) = j]$ in (2) and (3), we obtain β_j. As mentioned, the random variable $Q_m(t_i)$ forms a discrete-state Markov chain. Since the state transitions take place only at the arrival epochs, the system state $\{Q_m(t_i)\}, m = 1 \dots M$, also forms a discrete-state Markov chain. Having defined the formula for β_j, the state transition probability matrix of the chain is determined. Fig. 4 illustrates the system chain. The states with dotted lines are unobservable states.

3.2 Transition Probability Matrix Computation

Although we are able to obtain the transition probability matrix of the Markov chain in the general form, it is challenging to compute the matrix with inter-departure intervals following a general distribution $g(x)$. Several approaches are available to cope with this computation, including simulation such as Monte-Carlo, reinforcement learning such as Q-learning, and function approximation. Due to space constraints, we present here only the

8. The departure processes are independent as long as the paths are independent.

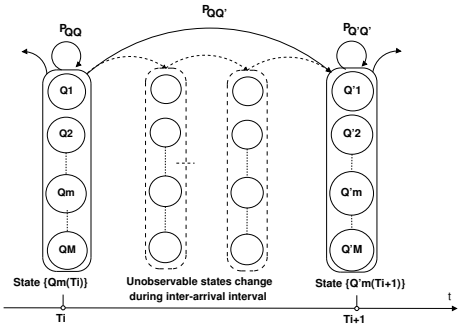

Fig. 4. The Imbedded Markov Chain.

function approximation approach. In particular, $g(x)$ is approximated by the a-stage Erlang distribution with rate ρ:

$$g(x) \approx \frac{\rho^a x^{a-1} e^{-\rho x}}{(a-1)!}$$

In Sec. 6.1, we show that the approximation is acceptable by studying RTT jitter of real heterogeneous networks. Since

$$\int_0^t \frac{\rho^{m+1} u^m e^{-\rho u}}{m!} du = 1 - \sum_{n=0}^{m} \frac{(\rho t)^n e^{-\rho t}}{n!}$$

$P[N(t) = j]$ can be computed directly:

$$\begin{aligned} P[N(t) = j] &= \frac{\rho}{a} \int_0^t \left\{ \sum_{m=ja-a}^{ja-1} - \sum_{m=ja}^{ja+a-1} \right\} \frac{(\rho u)^m e^{-\rho u}}{m!} du \\ &= \frac{1}{a} \sum_{n=ja}^{ja+a-1} (ja + a - n) \frac{(\rho t)^n e^{-\rho t}}{n!} \\ &+ \frac{1}{a} \sum_{n=ja-a}^{ja-1} (n - ja + a) \frac{(\rho t)^n e^{-\rho t}}{n!} \end{aligned}$$

Having obtained $P[N(t) = j]$, we can compute β_j in each scenario. In the first scenario, by assumption, bins arrive evenly at the traffic distributor with a constant rate λ. Subsequently, $t = t_{i+1} - t_i = 1/\lambda$. By substituting into the above $P[N(t) = j]$ formula, β_j is computed as follows:

$$\begin{aligned} \beta_j = P[N(t) = j] &= \frac{1}{a} \sum_{n=ja}^{ja+a-1} (ja + a - n) \frac{(\rho/\lambda)^n e^{-\rho/\lambda}}{n!} \\ &+ \frac{1}{a} \sum_{n=ja-a}^{ja-1} (n - ja + a) \frac{(\rho/\lambda)^n e^{-\rho/\lambda}}{n!} \end{aligned} \quad (4)$$

In the second scenario, by assumption, bins arrive at the traffic distributor with inter-arrival time following the exponential i.i.d random variable governed by p.d.f. $\lambda e^{-\lambda t}$. As a consequence, β_j can be computed as follows:

$$\begin{aligned} \beta_j &= \int_0^\infty P[N(t) = j] \lambda e^{-\lambda t} dt \quad (5) \\ &= \frac{1}{a} \left[\sum_{n=ja}^{ja+a-1} \frac{(ja + a - n)(\frac{\rho}{\lambda})^n}{(1 + \frac{\rho}{\lambda})^{n+1}} + \sum_{n=ja-a}^{ja-1} \frac{(n - ja + a)(\frac{\rho}{\lambda})^n}{(1 + \frac{\rho}{\lambda})^{n+1}} \right] \end{aligned}$$

Given the ρ/λ ratio, β_j can be directly computed from the above formula. Obtaining β_j for each queue, the state transition probability matrix of the Markov chain is fully determined. In real implementations, ρ/λ can be periodically estimated from the evolution of the system queues. In Sec. 7.5, we show that the control mechanism is robust to a certain level of errors in the ρ/λ estimation.

4 MDP Formulation

A time homogeneous Markov Decision Process (MDP) [34] consists of a four-component tuple $\{S, A, P, R\}$ in which $S = \{s\}$ is a countable state space; $A = \{A(s)\}$ is a finite action space where $A(s) = \{a\}$ is a set of admissible actions in state s; $R : S \times A(S) \rightarrow \Re_+$ is a non-negative immediate reward function that maps action a in state s to a non-negative value r; and $P = \{p(s'|s, a)\}$ is a set of conditional probabilities, in which $p(s'|s, a)$ is the probability that the system moves from state s to state s' if action $a, a \in A(s)$ is taken. In the unconstrained MDP formulation, the MDP has a single objective, which is to maximize the cumulative (average) reward achieved over a period of time. In the context of the problem under study, the objective is to minimize the time required to transfer an amount of data. Subsequently, the four-component tuple of the MDP is made of:

- S is the state space of the described system, which consists of $M, (M \geq 1)$ overlay paths. It is clear that S is the state space of the embedded Markov chain.
- $A = \{a_1, a_2 \dots a_M\}$ is the set of M possible actions, each of which corresponds to the action of forwarding a data bin to one of M overlay paths.
- P is the state transition probability matrix of the Markov chain, which can be calculated as shown in Sec. 3.2.
- R is the immediate reward function, which should reflect the minimum transfer time objective. According to Little's theorem [29], the average delay experienced by a bin is related to the average number of bins in the queue $\overline{D} = \frac{1}{\lambda} E[C_i]$. Therefore, we define the immediate reward function: $R(s, a) = \mu_a s + d_a$ where μ_a is the average inter-departure interval of the path chosen by action a; $s \in \{1, 2 \dots K\}$ is the state of the path; and d_a is the path propagation delay. In a practical implementation, d_a can be estimated as $0.5 \times \min(RTT)$. This estimation, however, does not undermine the generalization of the approach we propose.

A Markov policy is a description of behaviors, which specifies the action to be taken in correspondence to each system state and time step. If the policy is *stationary*, it specifies only the action to be taken in each state independently from the time steps. Given policy π and initial state s of the system, one can quantitatively evaluate π based on the expected cumulative reward, which is defined as follows:

$$v^\pi(s, T) \equiv E_s^\pi \left\{ \sum_{t=1}^{T} R(S(t), A_\pi(S(t))) \right\} \quad (6)$$

where $v^\pi(s, T)$ denotes the expected cumulative reward achieved by the decision maker from time step 1 to T with initial state $s, s \in S$; $R(S(t), A_\pi(S(t))$ denotes the immediate expected reward received by the decision

maker at time t by taking action $A_\pi(S(t))$ while the system is in state $S(t)$ in correspondence with policy π. Recall that the amount of data (in the number of bins) to be transferred is a random variable ν governed by a Geometric distribution with parameter $\gamma, (0 \leq \gamma \leq 1)$. Subsequently, the expected cumulative reward obtained by the decision maker when using policy π to transfer the data set is as follows:

$$v^\pi(s) \equiv E_s^\pi \left\{ \sum_{n=1}^{\infty} \sum_{t=1}^{n} R(S(t), A_\pi(S(t)))(1-\gamma)\gamma^{n-1} \right\} \quad (7)$$

Since the delay has a finite value, (7) can be futher simplified [34], which gives:

$$v^\pi(s) = R(s, a_\pi) + \sum_{s' \in S} \gamma p_a(s'|s) v^\pi(s') \quad (8)$$

where $R(s, a_\pi)$ is the immediate expected reward received by the decision maker when taking action a in state s in accordance with policy π for the case of discrete state space; and $p_a(s'|s)$ is the probability that the system moves from state s to state s' when action a is taken. In the problem under study, $p_a(s'|s)$ is computed as the product of corresponding β_j as shown in Sec. 3.2. An optimal policy is the one that minimizes the cumulative reward v^π. Note that, it is sufficient to find an optimal policy in the Markov policy space, since for any history-dependent policy, there is a Markov policy that yields the same cumulative reward. In this MDP formulation, it is possible to find a stationary optimal policy since $(0 < \gamma < 1)$ [34].

5 Optimal Data Distribution Algorithm

An optimal control strategy can be found by solving the formulated MDP. In principle, the standard dynamic programming techniques to solve an MDP (e.g. Value Iteration and Policy Iteration) can be used [34]. However, these techniques are computationally expensive, and in general, they are not suitable for solving the problem under study. Fortunately, by exploiting the system specificity, a significant fraction of the computation time can be saved. Therefore, we propose a new computationally efficient algorithm namely OPI, i.e. On-line Policy Iteration, to optimally distribute data over multiple paths. The OPI algorithm is designed based on the idea of the Policy Iteration algorithm and asymmetric dynamic programming. The following interesting observations are taken into account to increase the computational efficiency of the algorithm. First, dynamic programming and linear programming are computationally expensive, since they always sweep throughout the state and action spaces at every decision step to find an optimal action. The reason for such sweeping is that, in general, a system can move from one state to any other state, as a result of a taken action. Fortunately, in the system under study, the transition from one state to another is limited by physical constraints. For instance, it is impossible for a queue of hundreds of packets to be empty in a millisecond. As a result, we can greatly increase the computational efficiency by taking those constraints into account, i.e. only practically possible states and actions will be evaluated when searching for an optimal policy. Second, it is also common in practice that some states of a system are more frequently visited than others. These states are usually more important to the decision maker as they contribute more to the final outcome. Hence, instead of equally evaluating and improving the control policy for every state, we could spend more time on evaluating and improving the policy for those important states. This policy improvement strategy is known as the prioritized sweeping techniques, which guarantees an ϵ-optimal policy will be found if all states

Algorithm 1 Online Policy Iteration (OPI)

1: *// Initialize variables*
2: $\pi \leftarrow JSQ$ ▷ the starting policy is JSQ
3: $S = \{s_1, s_2 \dots s_M\}$, $s_i = \{1, 2 \dots K\}$ ▷ set of M queue
4: $A = \{1, 2 \dots M\}$ ▷ set of M actions
5: $R(s, a) = \mu_a s + d_a$ ▷ reward function
6: $\Lambda = \{\rho_1/\lambda \dots \rho_M/\lambda\}$ ▷ vector of path ρ/λ ratio
7: $\Theta = \{\theta_1 \dots \theta_M\}$, $\theta_i = \{\beta_1 \dots \beta_K\}$ ▷ vector of β_j
8: $v^\pi \leftarrow \emptyset$ ▷ vector of state/action values
9: $T_{sta} \leftarrow T$ ▷ period the system is stationary
10: $T_{jsq} \leftarrow T$ ▷ period the system uses JSQ
11: *// Following JSQ for a first few steps*
12: **while** $(T_{jsq} > 0)$ **do**
13: **if** $(bin == true)$ **then** ▷ a new bin arrives
14: $s \leftarrow S$ ▷ obtain the current state
15: $a \leftarrow \pi(s)$ ▷ obtain the corresponding action
16: $DitributeData(a)$ ▷ send bin to path a
17: $T_{jsq} \leftarrow T_{jsq} - 1$
18: **end if**
19: **end while**
20: *// Estimate the system model parameters*
21: **for** $i = 1, 2 \dots K$ **do**
22: $\rho_i/\lambda \leftarrow N_i^{depa}/N^{arvl}$ ▷ estimation in JSQ period
23: **end for**
24: $\Theta = ComputeBetaJ(\Lambda)$ ▷ get β_j for each path
25: *// On-line policy improvement*
26: **while** $done == false$ **do**
27: **if** $(bin == true)$ **then** ▷ a new bin arrives
28: $s \leftarrow S$ ▷ obtain the current state
29: $a \leftarrow \pi(s)$ ▷ obtain the corresponding action
30: $DitributeData(a)$ ▷ send bin to path a
31: $\Omega \leftarrow GetReachableStates(s, \Theta)$ ▷ from s
32: $v^\pi(s) \leftarrow R(s, a) + \sum_{s' \in \Omega} \gamma p_a(s'|s) v^\pi(s')$
33: *// Improve policy π for state s*
34: $\pi(s) \leftarrow \min_{a \in A} \{R(s, a) + \sum_{s' \in \Omega} \gamma p_a(s'|s) v^\pi(s')\}$
35: **else if** (π is NOT ϵ-optimal) **then**
36: *// Improve policy π for neighbor states*
37: **for** each $s \in \Omega$ **do**
38: $\omega \leftarrow GetReachableStates(s, \Theta)$
39: $\pi(s) \leftarrow \min_{a \in A} \{R(s, a) + \sum_{s' \in \omega} \gamma p_a(s'|s) v^\pi(s')\}$
40: **end for**
41: **end if**
42: **if** $(T_{sta} \leq 0)$ **then** ▷ re-estimate system model
43: **for** $i = 1, 2 \dots K$ **do**
44: $\rho_i/\lambda \leftarrow N_i^{depa}/N^{arvl}$ ▷ estimation in T_{sta}
45: **end for**
46: $\Theta = ComputeBetaJ(\Lambda)$ ▷ get β_j for each path
47: $T_{sta} \leftarrow T$
48: **else**
49: $T_{sta} \leftarrow T_{sta} - 1$
50: **end if**
51: **end while**;

of the system are visited infinitely often [34]. Third, dynamic programming, in particular Policy Iteration, is an incremental policy improvement process: the decision policy of the current step is statistically better than the policies of the previous steps. However, this improvement process is non-linear, i.e. a significant improvement is usually observed at some initial steps, followed by an incremental improvement in further steps. Thus, it is not necessary to wait until an optimal policy is found. A policy obtained after the first few steps is good enough in many cases. Furthermore, if a reasonable policy is used as the starting policy, the policy improvement process could quickly converge to an optimal policy.

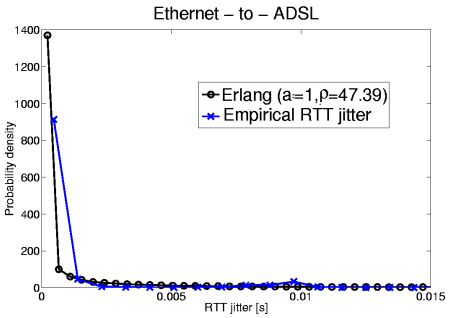

Fig. 5. PDF of RTT Jitter and Erlang in a real network.

The OPI pseudo-code is detailed in Algorithm 1. In brief, it works as follows: **Step 1**: select JSQ as the starting policy and start to distribute data immediately using this policy for a few dozens of data bins. Since JSQ is an optimal policy for the case of two symmetric queues, and an acceptable policy for several other cases [27], a lot of computation can be saved by starting from JSQ. **Step 2**: estimate ρ/λ ratio and compute β_j for each path. **Step 3**: when a new data bin arrives, observe the current system state, take an appropriate action following the current policy to distribute the data bin, and obtain the immediate reward for the action taken. **Step 4**: do the policy improvement for the currently observed state, and if the time is sufficient, for the neighboring states. During the improvement process, only reachable states will be evaluated. In the system under study, only states with sufficiently large β_j (e.g $\beta_j > 0.0001$) are reachable. Also note that this policy improvement will be carried out until an ϵ-optimal policy is obtained. Afterward, no further improvement is required. Therefore, OPI is more beneficial for a large data transfer (e.g. live content streaming, file sharing ...). **Step 5**: if the stationary period T has expired, go to Step 2 to update the system model. Otherwise, go to Step 3. Since the OPI algorithm is a special case of the Policy Iteration algorithm where the policy improvement step is done on the fly, the convergence of the algorithm is proved when $0 < \gamma < 1$. Details of the proof can be found in [34].

5.1 Complexity Analysis of OPI

In general, the OPI algorithm is lightweight and can be used for making decisions on the fly. More specifically, at each decision epoch, the decision is made instantly by mapping an observed system state to the current control policy. In case an ϵ-optimal policy has not been obtained, only (8) has to be computed, the complexity of which is $O(N)$, $(N = sizeof(\Omega))$ for each $a \in A$ (Ω is the set of reachable states from a state $s \in S$). In the problem under study, Ω is significantly reduced by the path capacity and the data arrival process. Therefore, N is usually in the range of hundreds to thousands, depending on the number of transmission paths M. For instance, in our study, $N = 25 \ldots 100$ for 2 paths, $125 \ldots 1000$ for 3 paths, $725 \ldots 10000$ for 4 paths.

6 OPI AT WORK

To evaluate the performance of the proposed approach, a simulation study using *ns-2* was conducted. In this section we show the comparison of OPI with WRR and JSQ, in network settings that allow to obtain the highest performance gain [42], while simulations in different network configurations are presented in Section 7. However, before digging into the details of the performance analysis, in Sec. 6.1 we analyze the RTT jitter to verify the approximation reported Sec. 3.2.

6.1 RTT Jitter Distribution

To justify the function approximation approach adopted in Sec. 3.2, we studied the distribution of the RTT jitter we measured over real heterogeneous network paths. The results provide an experimental proof of the hypothesis for which such jitter is well approximated by an Erlang distribution (Sec. 3.2). The network scenarios on which the data was collected are part of a real heterogeneous wired/wireless testbed (see [7] for more details). To show our claim, we examined network paths with different characteristics. In particular, the following four types of paths were considered: (i) Ethernet-to-ADSL; (ii) Ethernet-to-802.11b; (iii) 802.11b-to-802.11b (ad-hoc mode); and (iv) 802.11b-to-802.11b (infrastructure mode). By taking into account RTT jitters related to various types of paths with different characteristics in both wired and wireless environments, we justify our assumption in more general and real network scenarios. To provide a graphical evidence of this assumption in Fig. 5 we report the RTT jitter distribution for one of the above mentioned network scenarios. To provide a numerical evidence, we evaluated the discrepancy (by using the λ^2 [32]) between the empirical RTT jitter and three analytical distributions (Erlang, Normal, and Weibull). The obtained values, reported in Table 1, show that the Erlang distribution provides a good fit in all considered scenarios.

TABLE 1
Discrepancy measures of Erlang, Normal, and Weibull.

Type of path	Erlang	Normal	Weibull
Ethernet-Adsl	0.22	1.60	0.46
Ethernet-802.11	0.44	1.24	0.54
802.11-802.11(Ad-hoc)	0.20	5.15	0.74
802.11-802.11(Infrastructure)	0.25	3.89	0.48

6.2 Minimal-Time File Transfer

The network we used for the simulations was constructed in the following way: i) we took the topology of the Telstra's backbone; ii) we used the values of the link delays from Rocketfuel[9]; iii) we chose different capacity values for the various links to simulate different operating conditions. The resulting network is illustrated in Fig. 6 with the capacity and propagation delay of each link. Background traffic injected into each link was generated using a random number of FTP, CBR and ON/OFF traffic sources, the parameters of which, e.g. file size, sending rate, and on/off time were chosen randomly to create a certain level of variation in the overlay path performance (see Table 2 for more details). Fig. 7 shows the dynamics of the background traffic through the evolution of the bottleneck link queue both packet-by-packet and averaged every 250ms.

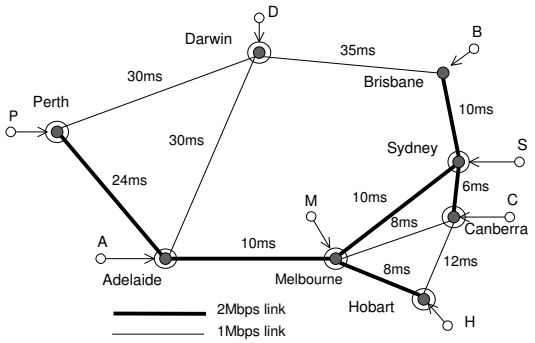

Fig. 6. Network topology used for the simulations.

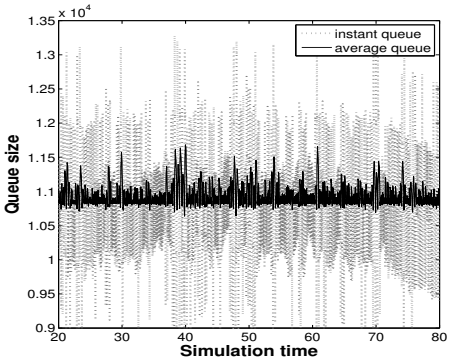

Fig. 7. Simulated background traffic.

TABLE 2
Parameters of simulated background traffic sources.

	FTP	CBR	ON/OFF
NumSrc per 100s	Rand[10-100]	rand[10-50]	rand[10-50]
Start time (s)	rand[0-100]	rand[0-100]	rand[0-100]
Duration (s)	rand[0-100]	rand[0-100]	rand[0-100]
Packet size (Bytes)	1500	500	1000
Rate(Mbps)	N/A	Rand[0.1-0.5]	Rand[0.1-0.5]
Data size	rand[10-1500]KB	rand[5-50]MB	N/A
Burst time	N/A	N/A	500ms
Idle time	N/A	N/A	500ms
Random/Shape	N/A	True (see *ns*-2)	Pareto(1.6)

9. http://www.cs.washington.edu/research/networking/rocketfuel

The performance of WRR[10], JSQ, and OPI algorithms was first compared in the following simulation scenarios:

Scenario 1: A multi-path capable application located at Sydney wants to transfer multimedia contents to its partner located at Perth using an MPRON. The application establishes 2 overlay paths, which are Sydney-Perth via Brisbane-Darwin, and Sydney-Perth via Melbourne-Adelaide. The contents are segmented into a stream of 1500-Byte bins and they are dumped into the application traffic distributor at a constant rate of 50 bins per second.

Scenario 2: A multi-path capable application located at Canberra wants to transfer multimedia contents to its partner located at Melbourne using an MPRON. The application establishes 3 overlay paths, which are Canberra-Melbourne direct route, Canberra-Melbourne via Sydney, and Canberra-Melbourne via Hobart to transfer the contents. The contents are segmented into a stream of 1500-Byte bins and dumped into the traffic distributor at a constant rate of 100 bins per second.

Scenario 3: An application located at Adelaide wants to transfer multimedia contents to its partner located at Canberra using an MPRON. Since the application is not multi-path capable, it simply routes the contents to the MPRON relay located in Melbourne. The relay then forwards the received contents to Canberra using 3 overlay paths, which are Melbourne-Canberra direct route, Melbourne-Canberra via Sydney, and Melbourne-Canberra via Hobart. The contents arrive at the relay in a stream of 1500-Byte bins following the Poisson distribution with the average rate of 100 bins per second.

We use results of [42], claiming that most of performance gains are realized by using 2 to 4 overlay paths with an extra small amount of bandwidth. In addition, in real implementations, overhead introduced by a large number of path must be considered. We investigated the impact of the number of nodes in Sec. 7.2. In all the scenarios, the amount of data to be transferred was chosen between 10KB-100MB. Since the day-time traffic is usually characterized by a higher intensity and shorter transfer sessions in comparison with the night-time, we simulated the scenarios using the following background traffic settings: (i) *Daytime setting*, the background traffic occupies $50\%-70\%$ of the path capacity, and the data size subsequently is 10KB, 50KB, 100KB, 500KB, 1MB, 5MB, and 10MB; (ii) *Night-time setting*, the background traffic occupies $30\% - 50\%$ of the path capacity, and the simulated data size is 50MB, and 100MB. The transport layer protocol in use was TCP. Unless otherwise specified, the simulation results are 10-run average, and presented in the form of the mean $\pm$ standard deviation. Tables 3, 4, and 5 show the transfer time obtained by each algorithm in the three simulated scenarios. In Fig. 8 we also report the results normalized over those obtained by WRR for visual comparison. As shown, OPI outperforms both WRR and JSQ in all considered scenarios. In particular, OPI performs better than WRR and JSQ by 15% ca.

10. Data bins are distributed proportionally to the path throughputs.

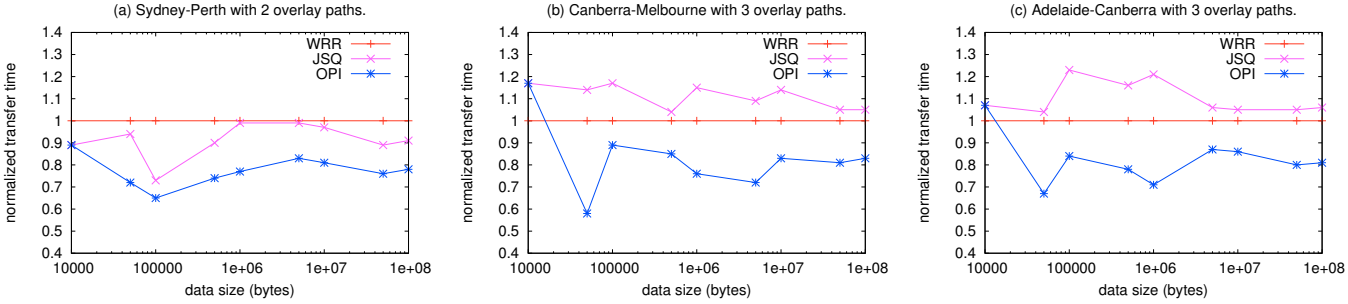

Fig. 8. Performance comparison of WRR, JSQ and OPI.

in Scenario 1, and 20% in Scenarios 2 and 3. The OPI performance stabilizes after the warm-up period, which was 1MB ca. of transferred data. This indicates that OPI is capable of approaching an optimal policy within the first 600 iterations. We also see that the performance of the algorithms in Scenario 1 is less stable in comparison with that in Scenario 2 and 3, i.e. the variance of the transfer time is larger. We believe the reason is that the variation of the background traffic in Scenario 1 is higher since the paths are comprised of more physical links. Between Scenario 2 and Scenario 3, there is no significant difference in performance gain. This indicates that OPI works well with both constant-bit-rate and Poisson arrival processes.

TABLE 3
Sydney-Perth transfer time with 2 overlay paths.

Data size	WRR (s)	JSQ (s)	**OPI** (s)
10KB	1.73 ± 0.66	1.54 ± 0.28	1.54 ± 0.28
50KB	6.81 ± 3.25	6.38 ± 4.61	4.90 ± 1.09
100KB	10.23 ± 5.4	7.43 ± 2.54	6.62 ± 1.20
500KB	42.97 ± 13.56	38.71 ± 7.43	31.76 ± 7.40
1MB	83.04 ± 19.55	82.55 ± 17.71	64.18 ± 11.58
5MB	153.45 ± 19.40	152.39 ± 13.22	127.89 ± 7.47
10MB	259.24 ± 17.85	252.56 ± 14.53	209.10 ± 6.01
50MB	911.09 ± 21.17	811.91 ± 28.56	695.34 ± 19.02
100MB	1823.58 ± 48.39	1666.57 ± 41.69	1419.04 ± 29.25

TABLE 4
Canberra-Melbourne transfer time with 3 overlay paths.

Data size	WRR (s)	JSQ (s)	**OPI** (s)
10KB	1.05 ± 0.15	1.23 ± 0.12	1.23 ± 0.12
50KB	2.39 ± 0.41	2.73 ± 0.49	1.39 ± 0.13
100KB	3.15 ± 0.88	3.68 ± 0.74	2.79 ± 0.41
500KB	8.07 ± 1.30	8.36 ± 1.09	6.88 ± 0.77
1MB	17.14 ± 1.82	19.67 ± 2.75	13.09 ± 1.76
5MB	79.42 ± 5.96	85.72 ± 6.02	64.40 ± 5.03
10MB	141.74 ± 5.51	153.99 ± 8.27	127.39 ± 4.64
50MB	625.75 ± 17.07	681.93 ± 19.42	523.48 ± 18.77
100MB	1282.28 ± 38.28	1349.81 ± 40.87	1044.21 ± 27.17

TABLE 5
Adelaide-Canberra transfer time with 3 overlay paths.

Data size	WRR (s)	JSQ (s)	**OPI** (s)
10KB	1.07 ± 0.19	1.15 ± 0.09	1.15 ± 0.09
50KB	2.91 ± 0.19	3.03 ± 0.24	1.94 ± 0.17
100KB	4.26 ± 0.22	5.26 ± 0.27	3.57 ± 0.24
500KB	13.05 ± 0.18	15.14 ± 0.20	10.12 ± 0.13
1MB	28.56 ± 2.53	34.47 ± 4.69	20.32 ± 2.05
5MB	87.80 ± 4.79	93.11 ± 5.03	76.01 ± 4.87
10MB	185.22 ± 8.69	194.41 ± 8.74	159.47 ± 5.65
50MB	665.16 ± 13.86	697.57 ± 13.03	531.80 ± 10.90
100MB	1321.63 ± 20.56	1397.38 ± 26.18	1069.87 ± 23.75

6.3 Minimal-Distortion Streaming

In the context of live streaming, e.g. IPTV, the content is streamed using datagram protocols, e.g. UDP, and the objective is often to minimize audio-visual QoS parameters such as the *distortion*. In this section, we show that OPI, by changing the reward function, can also be used to minimize the distortion in multi-path streaming. In the streaming process, the two main factors that lead to the distortion are packet loss and delay, and therefore the reward function of the MDP is changed as follows:

$$R(s,a) = (\mu_a s + d_a)(1 - p_{s,a}^{loss})$$

where $p_{s,a}^{loss}$ is the probability that a bin is lost in state s.

To evaluate the video distortion caused by delay and loss different models have been proposed in literature [28]. They are typically codec dependent and complex, as they take into account the correlation between frames. In order to avoid the codec dependence and to simplify our evaluation, we introduce a notion of distortion rate (r_d) computed as follows:

$$r_d = \frac{\sum_{i=1}^{C_u} w_i}{\sum_{i=1}^{C_t} w_i}$$

where C_t is the number of bins transferred; C_u is the number of bins that could not be decoded due to losses or late arrivals (i.e. bins arriving after the decoding deadline); w_i is the weight of bin i, which indicates the importance of the bin in the decoding process. Such weight, while not considering the correlation introduced by some codecs, allows to qualitatively evaluate the video distortion, for which different frames may provide different contributions. We compared the algorithms in terms of distortion rate when streaming data from location Canberra to location Adelaide using 2 and 3 overlay paths. The overlay paths subsequently are Canberra-Adelaide direct route, Canberra-Adelaide via Sydney-Melbourne, and Canberra-Adelaide via Hobart-Melbourne. The content is streamed at a rate of 100 bins per second, with bin size of 500 Bytes, and using UDP. The simulation lasts for 1000 seconds. Fig. 9 depicts the average distortion rate obtained by using of the three distribution algorithms over 30 runs. As depicted, OPI outperforms WRR and JSQ in both the scenarios. In the case of 2 paths, OPI reduced the distortion rate of 15% ca. compared to JSQ, and of 23% ca. compared to WRR.

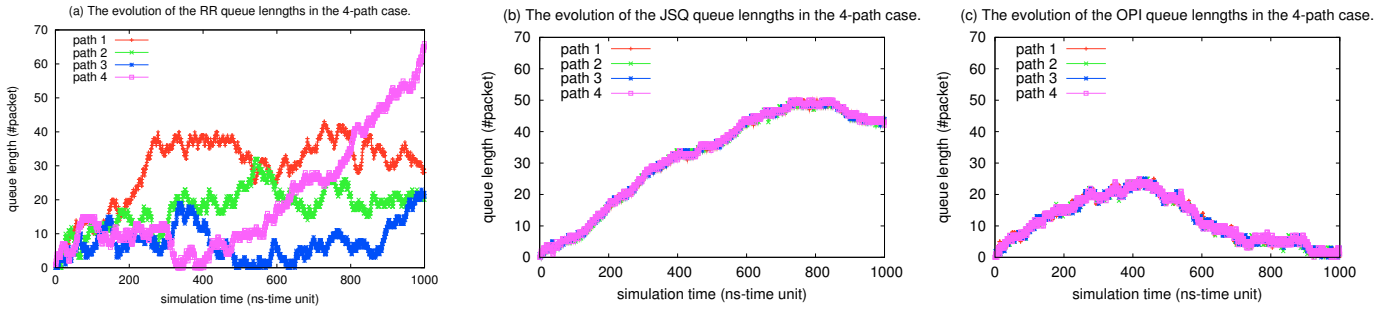

Fig. 10. The average path queue lengths (4 paths).

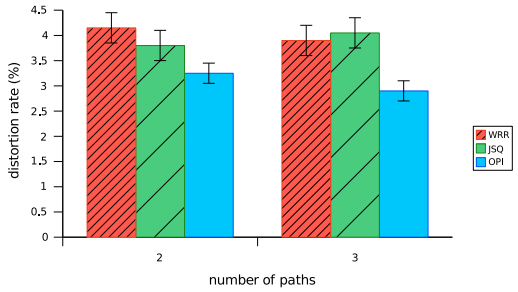

Fig. 9. Average distortion rate using 2 and 3 paths.

In the case of 3 paths, OPI has even a better performance gain, which is 28% and 26% in comparison with JSQ and WRR respectively.

6.4 Minimal-Delay and Loss

Apart from the the minimization of the file transfer time and of the distortion when streaming multimedia contents, we are also interested in the impact of the proposed multi-path control mechanism on QoS parameters such as end-to-end loss pattern, delay, and jitter. To provide a clear evidence of the benefits of the proposed approach on QoS parameters, we use the same reward function of Sec. 6.3 but, instead to work on the network reported in Fig 6, we investigate such impact in wireless environments, where the QoS parameters are more fluctuating. For this analysis, we consider a scenario in which two wireless end points, i.e. cars with wireless capabilities, exchange data over respectively 2, 3, and 4 wireless paths while moving in parallel in a [1500m x 1500m] flat grid area, with speed of $5-10m/s$. With respect to Fig. 2, here the *traffic distributor* module is integrated into the end points. Each wireless path is comprised of 3 wireless nodes (2 hosts and an Access Point), which implement Ricean radio propagation model[11], 802.11 MAC with the basic rate of 1Mbps, $600m$ of transmission range, and supporting Dynamic Source Routing Protocol [21]. To create disjoint wireless paths, different physical channels are used. Since in ns-2 the channels are orthogonal, the interference effect by default is not considered. To appropriately simulate transmission errors on 802.11 links, we implement a modification of the 802.11 error model[12] in conjunction with the Ricean fading propagation model. The two end nodes exchange data in a 500-Byte packet stream at the average rates of 2.0Mbps to 4.0Mbps depending on the number of paths in use. The stream of packets arrives at the traffic distributor following a Poisson distribution. The packets are then routed to the paths for transmission using 3 algorithms: OPI, WRR, and JSQ. The average loss rate (experienced by WRR) approximately is 6.0%. Table 6 shows the average, and variance in parenthesis, of delays and loss rates obtained by the algorithms with 2, 3 and 4 paths.

TABLE 6
Average (and Variance) of Delay and Loss Rate.

	2 paths		3 paths		4 paths	
	D(ms)	L(%)	D(ms)	L(%)	D(ms)	L(%)
WRR	7.7(2.4)	6.0(0.1)	7.4(2.3)	5.8(0.1)	7.6(2.1)	5.9(0.1)
JSQ	8.3(0.9)	6.2(0.1)	8.1(0.7)	6.5(0.1)	8.2(0.7)	6.1(0.1)
OPI	7.2(0.6)	5.9(0.1)	6.7(0.5)	4.7(0.1)	5.9(0.5)	4.2(0.1)

As shown in Table 6, OPI obtains the best performance. Compared to JSQ and WRR, OPI achieved smaller average end-to-end delay and loss rate in all considered scenarios. A good explanation for these results is in Fig. 10, which illustrates the evolution of the path queue lengths of the three algorithms obtained with 4 paths. As shown, the fluctuations of the WRR queues indicate the dynamics of the wireless channels, which reflect both congestion and transmission errors. These fluctuations result in high end-to-end delay variance and jitter (as shown in Fig. 11). In comparison with WRR, JSQ managed to maintain a stable and equal queue length for all paths. However, with this algorithm the queue lengths are high, which subsequently yields the high average end-to-end delay and loss rate. Compared to those obtained by WRR, the JSQ jitter is lower (Fig. 11). As for the evolution of the OPI queues, Fig. 10 clearly shows the online policy improvement process. In the warm-up period (approximately first 500KB of data), the evolution of the OPI queues is almost the same as the evolution of the JSQ queues since the two policies are, in fact, the same. However, after OPI obtained an optimal policy, its queue lengths start to decrease. As a result, OPI gives the lowest average end-to-end delay, the lowest delay variance as well as smaller and more stable jitter (Fig. 11). Besides the end-to-end delay character-

11. http://www.ece.cmu.edu/wireless/
12. http://www.comp.nus.edu.sg/~wuxiucha/research/reactive/

istics, the simulation results also indicate that the OPI algorithm improves end-to-end loss characteristics. Fig. 12 shows the distribution of consecutive losses obtained with OPI, WRR and JSQ. As shown, the probability of suffering from 2 consecutive losses is 0.07 for OPI while this probability is subsequently 0.12 and 0.20 for JSQ and WRR. The probability of having more than 2 consecutive losses is 0.01 for OPI, 0.07 for JSQ and 0.10 for WRR. Although our approach is built to pursue the improvement of delay and loss characteristics, it is also interesting to look at the behavior of another relevant QoS parameter such the throughput. Fig. 13 shows the average throughput obtained by the algorithms using 2 paths with the average data rate of 2Mbps. We can see the impact of the fading effect on the JSQ algorithm, which gives a low and fluctuating throughput. Due to the fading effect, throughput of the wireless channels changes rapidly. As a result, greedy strategies like JSQ suffer heavily from these rapid variations. On the other hand, the OPI algorithm, which adapts to the network conditions, obtains a significantly higher throughput. The WRR algorithm, which distributes packets equally on all paths also obtains a good throughput. However, it suffers more losses and high end-to-end delay variations since it does not take the channels dynamics into account. Finally, we can conclude that OPI reaches the best performance for all QoS parameters.

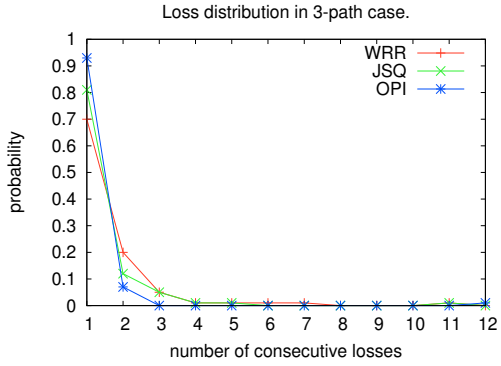

Fig. 12. Probability of consecutive losses.

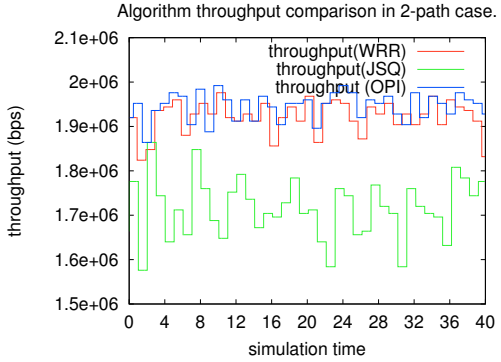

Fig. 13. Throughput computed in 1s interval (2 paths).

6.5 Load share analysis

In the previous simulations, we observed that in the long-term, OPI achieved a load share that was inversely proportional to the mean delay of the path. In particular, if the mean delay of each overlay path is respectively denoted as $\widehat{d}_1 \ldots \widehat{d}_M$ for $M (M > 1)$ overlay paths, then the portion of load share l_i of path i obtained in the long term by OPI is computed as follows:

$$l_i = \frac{\prod_{j=1, j\neq i}^{M} \widehat{d}_j}{\sum_{i=1}^{M} \prod_{j=1, j\neq i}^{M} \widehat{d}_j}$$

This observation indicates that OPI does maintain some level of fairness in the long term. However, it does not mean that one can "simulate" OPI by distributing data proportionally to the mean delay since (i) OPI does not maintain this fairness in the short-term; (ii) the mean delay is a function of the data distribution policy.

7 Further Investigations

In order to compare OPI with WRR and JSQ, in previous sections we have shown the performance of such algorithms in network settings that allow to obtain the highest performance gain [42]. In the following sections we investigate what happens in different settings, when an higher number of nodes and paths are used, when the paths are not independent, and when the path states are not accurately estimated. We also report preliminary results obtained on real networks using a prototype.

7.1 Impact of the number of nodes

We investigate the impact of the number of nodes composing the paths in both wireless and wired scenarios using the topology shown in Fig.14. The dotted lines in such topology represent a varying number of nodes on the three paths, which allows to test the effectiveness of OPI on paths comprising up to 10 nodes. Cross traffic is generated between all the hosts in the top and bottom of the topology with parameters reported in Tab.2. In Fig.15 we report the results obtained with 10 nodes per path. As we can see, OPI outperforms the other distribution algorithms in all the cases, allowing to obtain lower transfer times also in dense overlay networks with a high number of nodes and comprising wireless links.

7.2 Impact of the number of paths

We considered an amount of 10Mb of data to subsequently transfer at a fixed rate of 384Kbps over 1 to 10 homogeneous paths using the OPI algorithm. The capacity of each path is 0.1Mbps, and the propagation delay is 50ms. For comparison purposes, the transfer time obtained in each case (depicted in Fig. 16) was normalized over the transfer time of the 1-path case.

As shown, subsequently 50% to 85% of the performance gain is realized with 2 to 5 overlay paths. A further increase of the number of paths slightly increases the performance gain. This result supports the statement of [41]. In addition, in real implementations a large number of paths implies a high overhead.

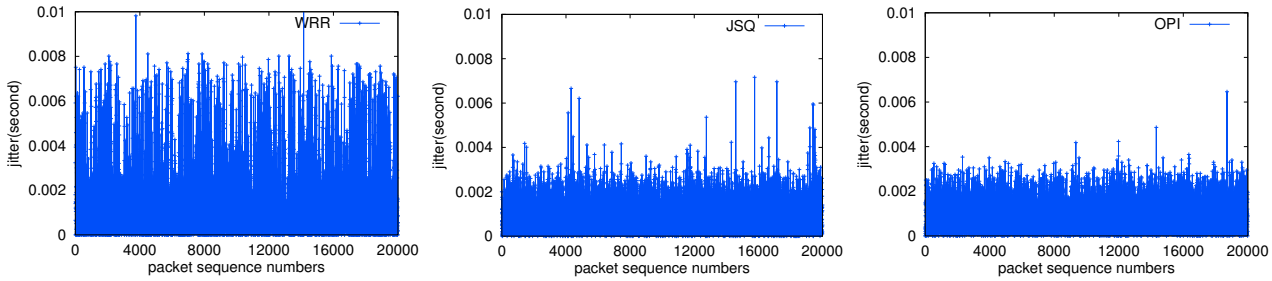

Fig. 11. Jitter obtained with *WRR (mean: 1.8ms, stdev: 1.6ms)*, *JSQ (mean: 0.78ms, stdev: 0.56ms)* and *OPI (mean: 0.61ms, stdev: 0.45ms)*, over 3 paths.

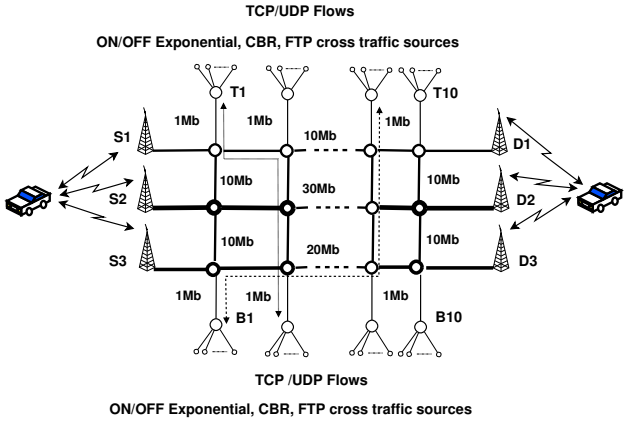

Fig. 14. Heterogeneous Network Topology.

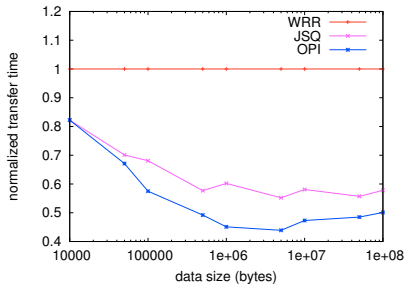

Fig. 15. Transfer time over paths composed of 10 nodes.

7.3 Impact of the path diversity

Considering the topology reported in Fig. 6, an amount of 10Mb of data is transferred from Canberra to Adelaide using 3 overlay paths, which subsequently were Canberra-Adelaide direct route, Canberra-Adelaide via Sydney, and Canberra-Adelaide via Hobart. These 3 paths share the common Melbourne-Adelaide link, the capacity of which is 2Mbps. We repeated the transfer with different source rates increasing from $0.5-4$Mbps to gradually saturate the shared link. The obtained performance of each algorithm was then compared with their performance when transferring the data at the same rate over 3 independent paths with an equivalent capacity and propagation delay. The result is illustrated in Fig. 17, which shows the percentage of performance loss due to the impact of the path dependence.

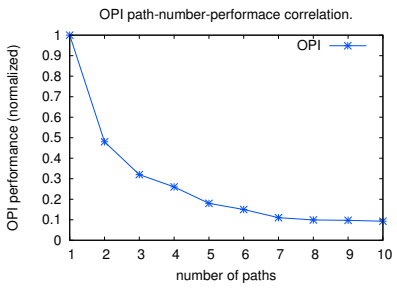

Fig. 16. Transfer time vs. number of paths.

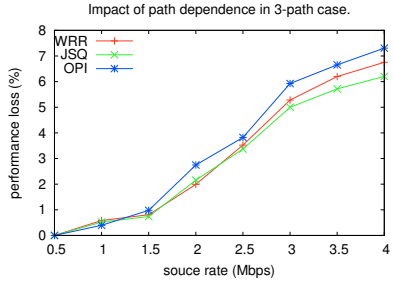

Fig. 17. Impact of the path diversity.

As shown in Fig. 17 the dependence of the overlay paths has a negative impact on performance of all the three algorithms. However, when the source rate was over 2Mbps, OPI suffered more since the assumption on the independence of the overlay paths was heavily violated. In average, the performance loss due to this phenomenon is approximately 5%. Nonetheless, the loss only becomes significant when the shared link is saturated, i.e. the source rate is equal to the shared link throughput (approximately 1.5Mbps in this case). This indicates that overlay paths can be considered independent if they share non-saturated links. In this situation, the performance loss is less than 1%.

7.4 Impact of the source rate

We repeated the simulation scenarios 1 and 3 with the source rate increasing from 240Kbps to 2.4Mbps. The amount of data transferred was 5MB. Fig.18 illustrates the obtained results in the case of 2 paths. As expected, the transfer time obtained with the three algorithms increases with the increase of the source rate. More interestingly, we can see that the OPI algorithm reached the saturation point quicker than the JSQ and the WRR. This implies that OPI has a better bandwidth utilization, i.e. with the same source rate, OPI distributes the load

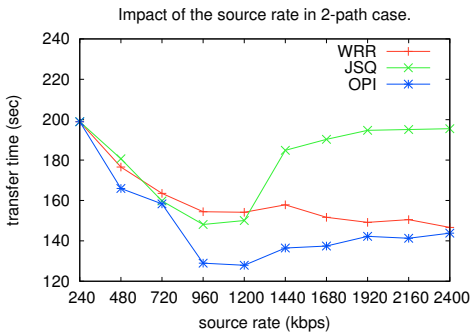

Fig. 18. Impact of the source rate.

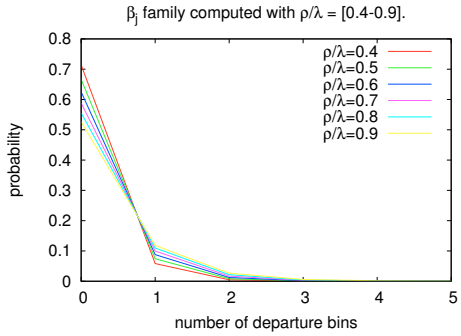

Fig. 19. β_j Family.

in a more optimal way utilizing the available bandwidth of the paths. Meanwhile, both WRR and JSQ require a higher source rate to saturate all the paths indicating that some paths were saturated before the others. As a consequence, OPI achieves a higher throughput, and therefore, a lower transfer time (this is also confirmed by the results related to the impact of OPI on QoS parameters reported in Sec. 6.4). We also observe that a further increase of the source rate even leads to the decrease of the performance of the JSQ and OPI. This is because, when the source rate is considerably higher than the path throughput, the queues of the path state monitoring mechanism are always full. As a result, the JSQ algorithm operates exactly like the simple RR algorithm. Since the two paths in simulation scenario 1 were asymmetric, distributing data equally on each path gave a significantly longer transfer time as shown by the JSQ. In a similar manner, when the queues were full, the OPI algorithm distributed data almost proportionally to the ρ/λ ratio of each path. As a consequence, the performance of the OPI was close to the performance of the WRR as illustrated in Fig. 18.

7.5 OPI Sensitivity and Robustness

Looking at the formulas used to compute β_j ((4) and (5) given in Sec. 3.2), we can see that the performance of OPI is only affected by the ρ/λ ratio, but not the absolute values of ρ and λ. Thus, as long as the ratio is maintained, the OPI performance is guaranteed. Moreover, OPI can resist to a certain level of error in the ρ/λ estimation. Fig.19 shows the values of β_j computed using different ρ/λ ratios. As shown, β_j values computed with ρ/λ ranging from $0.5 - 1.0$ are relatively close, which consequently would give a similar performance. To increase the performance of the algorithm, the ρ/λ ratio must be estimated as close to the actual value as possible. Since the ρ/λ ratio is an average value, a long enough history would give a better estimation. However, a trade-off between a long history and the dynamics of the overlay path should be found. In this study, the ρ/λ ratio is re-estimated after every 10MB of data transferred. However, if the stationary period of an overlay path is known, a better ρ/λ estimation can be obtained.

8 Experiments over Real Networks

We developed a first prototype of the traffic distributor to work in real network environments and we are currently using it on a testbed to evaluate the potential of multi-path traffic distribution algorithms aided by the considered path state monitoring approach. The prototype works by monitoring the status of the paths periodically (i.e. for every few tens of packets) because this simplifies the architecture and because our simulations showed that the benefits remain almost unchanged (see Sec. 3). Besides, the design and the implementation of the prototype are general enough to accommodate decisions on a per packet basis. In the following we present the outcome of a set of experiments performed configuring the network testbed to use two paths between sources and destinations, and introducing random values of loss and delay on the such paths. Such values were periodically changed to emulate time-varying network conditions. The testbed is composed of ten Linux-based routers (Intel P4 3.4 GHz, 2 GB RAM, 3 Fast-Ethernet 10/100 PCI interfaces, with Fedora Linux release 10) connected back-to-back, and it is configured to use two paths of five hops each, between the sources and destinations (notebooks equipped with Debian Linux release 5.0). We performed experiments by using a packet-level traffic generator called D-ITG[13], and emulating the behaviour of varying network conditions (delay $\in [20, 200]$ ms, loss rate $\in [0.01, 0.1]$) using NETEM[14]. We performed two different kinds of experiments, instructing the multi-path distributor to seek for lower delay in the first one and for lower losses in the second one. Fig. 20 shows the delay and jitter obtained in the first experiments and the throughput and packet loss obtained in the second one: experimental results confirm the simulation results shown in this paper. We report the ratio between the values obtained with OPI and WRR when using UDP and different packet rates. Each experiment has been performed 10 times, and the values in Fig. 20 represent the average value. The results show that the multi-path transmission using OPI allows to achieve lower delay, jitter and loss as well as higher

13. http://www.grid.unina.it/software/ITG
14. http://www.linuxfoundation.org/en/Net:Netem

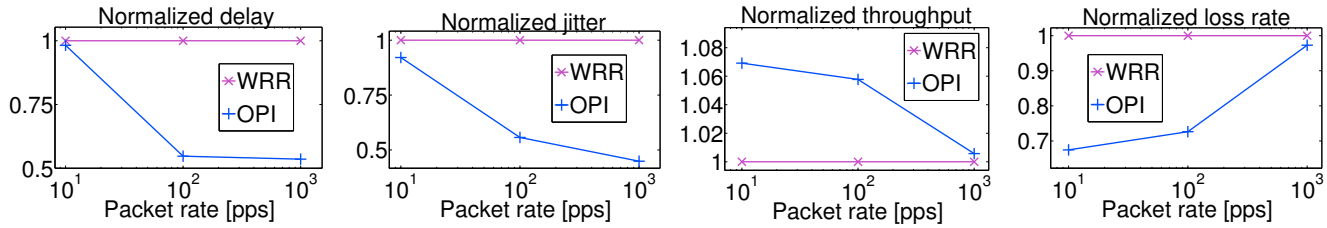

Fig. 20. A comparison between OPI and WRR over real networks.

throughput, effectively choosing the best available path. We can conclude that preliminary results indicate that our multi-path scheme is actually able to improve the performance of the communications over real networks.

9 Conclusion

In this paper we studied the problem of high performance multi-path data transfers in the context of multi-path routing overlay networks. We proposed a control mechanism based on Markov Decision Process to distribute traffic over multiple overlay paths while optimizing QoS metrics such as the transfer time, delay and losses. The optimal control problem was formulated as a Markov Decision Process, and a computationally efficient algorithm, named Online Policy Iteration, was used to optimally transfer the traffic by solving the formulated MDP on-line. Performance of the proposed control mechanism was evaluated and compared with the classical multi-path schemes including Weighted Round Robin and Join the Shortest Queue in both simulation and over a real testbed. The results indicate that the proposed control mechanism significantly outperforms classical multi-path schemes in all considered scenarios.

References

[1] A. Akella *et al.*, "A measurement-based analysis of multihoming," in *ACM SIGCOMM 2003*, Aug. 2003, pp. 353–364.

[2] ——, "A comparison of overlay routing and multihoming route control," in *ACM SIGCOMM 2004*, Aug. 2004, pp. 93–106.

[3] E. Altman, "Applications of markov decision processes in communication networks," in *Handbook of Markov Decision Processes: Methods and Applications*. Kluwer, 2002.

[4] D. Andersen *et al.*, "Resilient overlay networks," *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 131–145, Dec. 2001.

[5] D. G. Andersen, "Improving end-to-end availability using overlay networks," Ph.D., Massachusetts Inst. of Tech., Feb. 2005.

[6] C. Barakat *et al.*, "Modeling internet backbone traffic at the flow level," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 1–11, 2003.

[7] A. Botta *et al.*, "Identification of network bricks in heterogeneous scenarios," *Computer Networks*, vol. 52, no. 15, pp. 2975–2987, 2008.

[8] V. Bui *et al.*, "Improving multipath live streaming performance with markov decision processes," in *IEEE ISCIT 2007*, Oct. 2007.

[9] ——, "Modelling internet end-to-end loss behaviours: A new approach," in *Proc. of Asia Modelling Symposium 2007*, Mar. 2007.

[10] ——, "A game theoretic framework for multipath optimal data transfer in multiuser overlay networks," *IEEE ICC 08*, May 2008.

[11] ——, "An mdp-based approach for multipath data transmission over wireless networks," in *IEEE ICC 2008*, May 2008.

[12] C. Cetinkaya *et al.*, "Opportunistic traffic scheduling over multiple network paths," in *IEEE INFOCOM*, Mar. 2004, pp. 1928–1937.

[13] J. Chen *et al.*, "An efficient multipath forwarding method," in *IEEE INFOCOM*, Mar. 1998, pp. 1418–1425.

[14] W. C. Cheng *et al.*, "Large-scale data collection: a coordinated approach," in *IEEE INFOCOM 2003*, Mar. 2003, pp. 218–228.

[15] D. R. Cox, *Renewal Theory*. London, GB: Methuen & Co., 1962.

[16] S. Ganguly *et al.*, "Fast replication in content distribution overlays," in *IEEE INFOCOM*, Mar. 2005, pp. 2246–2256.

[17] Y. Hasegawa *et al.*, "Improved data distribution for multipath tcp communication," in *IEEE GLOBECOM '05*.

[18] J. He and J. Rexford, "Toward internet-wide multipath routing," *Network, IEEE*, vol. 22, no. 2, pp. 16–21, March-April 2008.

[19] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 951–964, 2006.

[20] J. R. Iyengar *et al.*, "Concurrent multipath transfer using transport layer multihoming: performance under varying bandwidth proportions," in *IEEE MILCOM*, Oct. 2004, pp. 238–244.

[21] D. Johnson *et al.*, *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, 2001, ch. 5, pp. 139–172.

[22] D. Jurca *et al.*, "Video packet selection and scheduling for multipath streaming," *IEEE Trans. Multimedia*, vol. 9, no. 3, pp. 629–641.

[23] T. Karagiannis *et al.*, "A nonstationary poisson view of internet traffic," in *IEEE INFOCOM*, Jan. 2004, pp. 177–187.

[24] L. Kleinrock, *Queueing Systems Volume 1: Theory*. New York, USA: John Wiley & Son, 1975.

[25] A. Konrad *et al.*, "Choosing an accurate network path model," in *ACM SIGMETRICS 2003*, Jun. 2003, pp. 314–315.

[26] S.-J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *IEEE ICC 2001*.

[27] J. Y.-T. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. USA: Chapman and Hall/CRC, 2004.

[28] Y. Liang, J. Apostolopoulos, and B. Girod, "Analysis of packet loss for compressed video: does burst-length matter?" in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[29] J. Little, "A proof of the queueing formula $l = \lambda w$," *Operations Research*, vol. 9, pp. 383–387, 1961.

[30] X. Liu *et al.*, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 10, pp. 2053–2064, Oct. 2001.

[31] Z. Ma *et al.*, "A new multi-path selection scheme for video streaming on overlay networks," in *IEEE ICC 2004*.

[32] S. Pederson and M. Johnson, "Estimating model discrepancy," in *Technometrics, 32(3)*, Aug. 1990, pp. 305–314.

[33] A. Pescapé *et al.*, "End-to-end packet-channel bayesian model applied to heterogeneous wireless networks," in *GLOBECOM '05. IEEE*, vol. 1, Dec. 2005.

[34] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. USA: John Wiley and Sons, 1994.

[35] R. S. Sutton *et al.*, *Reinforcement Learning: An Introduction*. USA: The MIT Press, 1999.

[36] S. Tao *et al.*, "Exploring the performance benefits of end-to-end path switching," in *IEEE ICNP 2004*, Oct. 2004, pp. 304–315.

[37] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker, "In search of path diversity in isp networks," in *ACM IMC '03*.

[38] E. Vergetis *et al.*, "Packet-level diversity - from theory to practice: an 802.11-based experimental investigation," in *MobiCom '06*.

[39] L. Vu *et al.*, "Measurement and modeling a large-scale overlay for multimedia streaming," in *ICST fQShine 2007*, 2007.

[40] S. Vutukury *et al.*, "Mpath: A loop-free multipath routing algorithm," *Journal of Microprocessors and Microsystems*, vol. 24, pp. 319–327, 2000.

[41] B. Wang *et al.*, "Multipath overlay data transfer," University of Massachusetts Amherst, Amherst, MA, USA, Tech. Rep., 2005.

[42] ——, "Application-layer multipath data transfer via tcp: Schemes and performance tradeoffs," *Journal of Performance Evaluation, Elsevier*, vol. 64, pp. 965–977, 2007.

[43] Y. Zhang *et al.*, "On the constancy of internet path properties," in *ACM SIGCOMM Workshop on Internet Measurement*, 2001.

Vinh Bui received his BE from Hanoi University of Technology, MSc and PhD from University of New South Wales - Australia in 2001 and 2008 respectively. He is currently with the Defence and Security Application Research Centre, University of New South Wales. His research interests include distributed systems, computer and communication networks, embedded systems and robotics. New South Wales, Australia. His research interests covers distributed systems, computer networks and network tomography.

Weiping Zhu received his B.Eng and M.Eng from Hunan University and China University of Mining and Technology in 1982 and 1984, respectively. He received Ph.D from University of New South Wales, Australia in 1992. He is currently a senior lecturer at the School of Information Technology and Electrical Engineering, University of New South Wales, Australia. His research interests covers distributed systems, computer networks and network tomography.

Alessio Botta is a Ph.D. in Computer Engineering at the Department of Computer Engineering and Systems of the University of Napoli Federico II (Italy). He received the M.S. Laurea Degree in Telecommunications Engineering in 2004 from the same University. His research interests fall in the area of networking, with specific regards to network monitoring and measurements. He is a member of the IEEE.

Antonio Pescape' is an Assistant Professor at the Department of Computer Engineering and Systems of the University of Napoli Federico II (Italy). He received the M.S. Laurea Degree (summa cum laude) in Computer Engineering and the Ph.D. in Computer Engineering and Systems, both at University of Napoli Federico II. As member of COMICS research group at University of Napoli Federico II and ITeM Lab of CINI (Italy), Antonio Pescape' is a principal investigator for several national and international research projects. His research interests are in the networking field with focus on models and algorithms for Internet Traffic, Network Measurements and Management of heterogeneous IP networks, and Network Security. Antonio Pescape' has coauthored over 80 journal and conference publications. He is an IEEE senior member and he has served and serves on several technical program committees of IEEE and ACM conferences (IEEE Globecom, IEEE ICC, IEEE WCNC, IEEE HPSR, ...). He also serves as Editorial Board Member of IEEE Survey and Tutorials and was guest editor for the special issue of Computer Networks on "Traffic classification and its applications to modern networks".