University of Brasília

Institute of Exact Sciences
Department of Computer Science

A Multi-agent Architecture Applying Trust and Reputation Over Unknown Partners for Live Video Distributed Transcoding in Open Environments

Charles Antônio Nascimento Costa

Thesis presented in partial fulfillment of the requirements for the degree of
Master of Science in Informatics

Supervisor
Prof. Dr. Célia Ghedini Ralha

Brasília
2022

University of Brasília

Institute of Exact Sciences
Department of Computer Science

A Multi-agent Architecture Applying Trust and Reputation Over Unknown Partners for Live Video Distributed Transcoding in Open Environments

Charles Antônio Nascimento Costa

Thesis presented in partial fulfillment of the requirements for the degree of
Master of Science in Informatics

Prof. Dr. Célia Ghedini Ralha (Supervisor)
CIC/UnB

Prof. Dr. Jaime Simão Sichman
Poli/USP

Prof. Dr. Mylène C. Q. Farias
CIC/UnB

Prof. Dr. Ricardo Pezzuol Jacobi
Coordinator do Post Graduation Program in Informatics

Brasília, February 16, 2022

Dedication

I dedicate this thesis to my wife Elaine and my daughters Érica and Helena.

Acknowledgements

I thank my family for the support, encouragement, and unconditional love throughout my life, particularly during the time I was dedicated to this work.

I would also like to thank my co-workers in The Chamber of Deputies, who never refused to help me when I needed it.

In particular, I thank Prof. Dr. Célia Ghedini Ralha, who believed in me and accepted me as a master's student, even though being 14 years apart from the university. I am also grateful for her availability and dedication with which she always attends to her advisees, even during weekends, holidays, and the troubled times we are experiencing.

I thank the University of Brasília, this special place, which will always be part of my life.

Resumo expandido

Projetistas de sistemas tem sido confrontados com aplicações e sistemas do mundo real que são inerentemente distribuídas e abertas. Um sistema inerentemente aberto é um no qual é impossível estabelecer controle global ou, também dizendo, aquele no qual uma única entidade não é capaz de possuir uma descrição completa do estado do sistema. Sistemas que atendem a essa descrição são complexos, e projetá-los é desafiante. Uma forma de lidar com esses desafios é abordar o problema como o projeto de um sistema multiagente. Um agente é um sistema computadorizado dotado de autonomia para agir em nome de seu proprietário. Um sistema multiagente é uma sociedade de agentes que interagem sob determinadas regras para alcançar metas comuns ou individuais. Um exemplo de problema complexo que poderia se beneficiar de uma abordagem multiagente é a distribuição de vídeo através da Internet.

Uma das razões para o crescimento rápido do consumo de dados na Internet é a crescente demanda por conteúdo em vídeo. Entre os provedores de *streaming* de vídeo ao vivo, a técnica *Streaming* de Vídeo Adaptativo (*Adaptive Bitrate Streaming - ABR*) se tornou o padrão de fato da indústria. ABR é uma forma conveniente de distribuir vídeo pela Internet para muitos usuários simultaneamente. Para descrever a técnica ABR brevemente, um *streaming* de vídeo é divido em segmentos que são transcodificados em diferentes taxas de *bits*, assim os usuários podem se adaptar, consumindo a representação que melhor se conforma com a sua largura de banda. Os recursos computacionais que a transcodificação demanda não são negligenciáveis. De fato, a transcodificação de vídeo representa custos relevantes para os provedores de vídeo ao vivo.

A bufferização empregada pelos *players* de vídeo compatíveis com a ABR é uma característica chave para determinar a previsibilidade das requisições de segmento de vídeo. Experimentos indicam que a audiência de vídeos pela Internet prefere representações com altas taxas de *bits*, sendo que constantes interrupções prejudicam a qualidade da experiência. Uma função de utilidade básica de uma sessão de vídeo pode ser definida como a razão entre a taxa de *bits* média, contrabalançada pela suavidade da reprodução. Suavidade da reprodução é a razão entre o tempo gasto esperando o *buffer* de vídeo ser

preenchido e o tempo total de exibição.

Em uma arquitetura baseada em nuvem, a periferia onde ficam os dispositivos dos usuários finais é chamada de Borda (*Edge*) ou Neblina (*Fog*). Desta forma, tirar vantagem desses recursos que estão geograficamente distribuídos é referenciado como Computação na Neblina (*Fog-Edge Computing - FEC*). O ambiente da FEC é definido como um complemento da núvem que emprega dispositivos na borda da rede para melhorar a qualidade de serviço através de um contínuo. Como um complemento da infraestrutura da Internet, o FEC herda algumas de suas características. O FEC tem muitos recursos computacionais ociosos, que estariam, teoricamente, disponíveis para serem utilizados entregando uma baixa latência. Usar esses dispositivos do FEC pode ser útil para a transcodificação distribuída de vídeo ao vivo. No entanto, a colaboração com dispositivos desconhecidos pode ser arriscada, pois não estão sob controle dos provedores ou dos usuários. Já que alguns dos nós do FEC tem autonomia deliberativa visando melhorar seu desempenho, nós podemos descrevê-los como agentes.

Uma sociedade composta de entidades autônomas, como um sistema multiagente, leva a possibilidade de uma parte destas entidades serem egoístas. Em outras palavras, é necessário saber em quem confiar. A aplicação de modelos de confiança e reputação é uma característica chave quando queremos lidar com o risco de delegar tarefas em ambientes abertos e semi-competitivos, tal como o FEC.

Para enfrentar a incerteza de colaborar com dispositivos no FEC, um agente racional A, antes de delegar uma tarefa da qual seu bem-estar depende para um agente B, precisa de alguma forma calcular a probabilidade de B completar a tarefa satisfatoriamente. Esta probabilidade representa o quanto o agente A avalia que B é digno de confiança quanto a tarefa em questão. De qualquer forma, um agente talvez não seja capaz de avaliar a confiabilidade de uma contraparte se eles nunca se encontraram antes. Uma solução recorrente para a falta de informação advinda de interação direta é perguntar a outros sobre a opinião que eles têm de um possível parceiro. A ponderação da confiança que uma comunidade deposita em um agente é chamada de reputação. Na literatura, há vários modelos de interação entre agentes baseados em confiança e reputação (*Trust and Reputation Models - T&RM*). Um dos aspectos que diferencia esses modelos são as fontes de informação que eles podem utilizar como insumo. No entanto, todos eles consideram a interação direta e/ou a opinião de testemunhas em seus cálculos.

Os algoritmos chamados de *Multi-Armed Bandits* (MAB) são aplicados quando um agente precisa escolher entre alternativas incertas. Agentes não sabem *a priori* qual é a distribuição de recompensas das escolhas postas à sua frente, mas têm certa confiança que existem escolhas melhores que outras. Os algoritmos MAB possuem duas fases, a fase de exploração e a fase de aproveitamento. Na fase de exploração são feitas escolhas para

tentar estimar a distribuição de recompensas de cada uma das opções testadas. Depois disso, o agente pode utilizar o conhecimento que adquiriu para escolher a melhor opção dentre as que passou a conhecer na fase de aproveitamento. Ao passar para a fase de aproveitamento, não queremos dizer que o agente sabe de forma incontestável qual é a melhor opção, já que a distribuição de recompensas verdadeira é ainda desconhecida e pode haver uma opção melhor dentre as que não foram escolhidas. Muitos algoritmos implementam diferentes estratégias para balancear exploração e aproveitamento. Para exemplificar, citamos *e-Greedy*, *e-First*, *e-Decreasing* e a família de algoritmos chamada Limites de Confiança Elevados (*Upper Confidence Bounds - UCB*).

Foram selecionados alguns trabalhos prévios que abordaram o problema de habilitar a transcodificação de vídeo ao vivo para dispositivos heterogêneos em ambientes distribuídos. Cada trabalho empregou um método específico, onde os autores validaram as abordagens em cenários distintos dificultando a comparação de desempenho dos mesmos. Assim, as soluções propostas foram analisadas procurando brechas onde modelos de confiança e reputação pudessem ser aplicados para trazer vantagens, tanto para os provedores quanto para os usuários finais. Destaca-se que os trabalhos pesquisados na literatura falham ao abordar ambientes abertos. No entanto, o problema da colaboração com agentes potencialmente maliciosos é proeminente quando se pretende empregar os dispositivos do usuário final. Seria interessante que as tarefas de transcodificação fossem designadas aos nós de forma dinâmica de acordo com o desempenho observado a cada turno de execução. Neste caso, o uso de uma métrica de confiança e reputação que represente uma avaliação geral da contribuição para a utilidade dos visualizadores, não apenas incluindo a estabilidade do nó, mas a competência em desempenhar a tarefa designada seria útil. Assim, uma proposta mais adequada ao problema poderia abordar três frentes: definir uma arquitetura baseada em agentes autônomos, capacitar a arquitetura a selecionar os nós apropriados para fazer a transcodificação em ambiente aberto e, ainda, avaliar a credibilidade de testemunhas evitando a influência de agentes não-confiáveis.

Como solução para o problema descrito, foram analisados os requisitos do sistema multiagente com a metodologia Tropos. Tropos é uma metodologia de desenvolvimento de software para programação orientada a agentes. Essa metodologia permite a representação de estados mentais como metas e qualidades. O aspecto que mais diferencia a metodologia Tropos de outras metodologias de desenvolvimento de software é a natureza centrada em agentes. A metodologia Tropos guia o desenvolvimento de soluções orientadas a agentes através de um conjunto de fases, pelas quais o desenvolvedor gradativamente vai refinando a representação do sistema. Da análise com a metodologia Tropos surgiu a proposta de uma arquitetura para transcodificação distribuída composto de agentes que desempenham três papéis: o Corretor (*Broker*), o *Proxy* do visualizador (*Viewer's proxy*)

e o Transcodificador (*Transcoder*). O *Proxy* do visualizador é o papel para os agentes que representam a audiência do *stream* de vídeo ao vivo. Esse papel é destinado aos agentes que requerem ao Corretor a adaptação do *stream* em ABR e interage com ele para avaliar o desempenho dos transcodificadores. O Transcodificador é o papel a ser desempenhado pelos agentes interessados em receber tarefas de transcodificação e serem recompensados por elas. A responsabilidade dos corretores é gerenciar a associação entre os *proxies* dos visualizadores e os transcodificadores para o benefício de ambos.

Pensando sobre o trabalho que os corretores desempenham no modelo proposto, em certo ponto eles irão formar um conjunto de transcodificadores dentre os quais alguns são bem conhecidos, enquanto outros não terão sido testados. Então, corretores devem balancear suas estratégias entre aproveitar os mais bem conhecidos ou explorar os desconhecidos para aprender sobre o desempenho deles. Aprender sobre os transcodificadores disponíveis, nós queremos dizer que os corretores devem formar uma crença sobre o quão bom transcodificador é um nó específico, com a ajuda da avaliação de um determinado grupo de visualizadores. Esta crença externa (relação não reflexiva) é uma medida da reputação do transcodificador na comunicade de visualizadores. Para o corretor, a reputação de uma transcodificador é representado por um par de valores: a confiabilidade do transcodificador e uma medida da confiança que se tem no primeiro valor, a credibilidade da confiança.

Para que o corretor tenha a capacidade de selecionar os nós de acordo com as regras estabelecidas foi introduzido o algoritmo ReNoS - *Reputation Node Selection*. O algoritmo foi projetado para balancear exploração e aproveitamento de forma que o nó mais confiável não seja sobrecarregado. Quando um novo transcodificador é registrado, recebe uma avaliação de confiança acima do limiar de cooperação e um pouco abaixo da maior avaliação possível, assim aumentando as chances de ser selecionado na próxima iteração. Um problema detectado com o uso do ReNoS é que ele requer que o valor de confiança inicial seja alto. Isto significa que, para usar o algoritmo, o agente que usa a confiança deve acreditar que um nó novo e desconhecido é tão bom quanto um muito conhecido e bem avaliado. De outra forma, a exploração não irá funcionar adequadamente. Esta política é semelhante a utilizada no algoritmo UCB1, onde as opções menos selecionadas até o momento são aquelas com as maiores chances de serem selecionadas no próximo turno. Para contornar esse problema, foi elaborada uma nova versão do algoritmo denominado ReNoS-II. O ReNoS-II é baseado na ideia do algoritmo conhecido como *Thompson Sampling*. Quando um novo transcodificador se registra recebe um valor de reputação com baixa confiança e credibilidade. Desta forma, a expectativa para a curva de desempenho é achatada e larga, semelhante a uma distribuição uniforme. Mas a medida que o transcodificador é testado e mais conhecimento se acumula sobre ele a credibilidade cresce e a

curva se estreita em torno do valor da confiança.

Para validação da arquitetura proposta foi realizado um experimento com o objetivo de verificar se a abordagem trata adequadamente o problema da transcodificação distribuída com nós do FEC. Foi utilizado um protótipo implementado seguindo estritamente as diretrizes da arquitetura, capaz de desempenhar as tarefas necessárias para distribuir a transcodificação em tempo real. Validar o modelo proposto que combina MAB e T&RM para selecionar nós no FEC envolve identificar as condições nas quais as características do ambiente FEC poderiam prejudicar as garantias dos algoritmos MAB. Uma dessas condições é quando os agentes não são verdadeiros em seus relatórios. Já que transcodificadores estão interessados em receber o maior número de tarefas de transcodificação possível, os nós não-confiáveis podem formar uma coalisão com visualizadores para tentar manipular as escolhas do corretor. Desta forma, o experimento inclui dois cenários distintos. No Cenário 01, o objetivo é obter uma linha base de comparação onde os agentes envolvidos não recusam interações sendo sempre verdadeiros nas trocas de informação. No cenário 02, o objetivo é observar o que acontece quando um transcodificador tenta manipular a transcodificação distribuída com ataques de relatórios falsos. Nesse experimento, a métrica utilizada para comparação foi o valor da recompensa acumulada pelo corretor ao longo de uma sessão de transcodificação. O experimento revelou que quando o algoritmo UCB1 foi empregado houve um decréscimo significativo do Cenário 01 para o Cenário 02. No entanto, não foi observado o mesmo decréscimo quando os algoritmos empregados foram ReNoS e ReNoS-II associados ao modelo FIRE. UCB1 e ReNoS produziram resultados semelhantes em termos de recompensa acumulada. Por outro lado, os resultados obtidos com o algoritmo ReNoS-II foram significativamente maiores do que os obtidos com UCB1 e ReNoS nos dois cenários, apesar da variância ter sido maior.

Pelos resultados dos experimentos realizados, conclui-se que o modelo proposto combinando MAB e T&RM para selecionar nós no FEC é promissor para aplicação no mundo real. Os resultados experimentais do algoritmo ReNoS se apresenta tão performativo quanto UCB1. O algoritmo ReNoS-II apresenta um desempenho melhor que o ReNos e UCB1 nos dois cenários testados. Enfim, os experimentos mostraram que ponderando e filtrando informação dos relatórios baseando-se na credibilidade das testemunhas é possível proteger o sistema de transcodificação distribuída no FEC de agentes não-confiáveis, evitando danos causados pela formação de coalisões.

Palavras-chave: Sistemas multiagentes, transcodificação distribuída, modelos de confiança e reputação

Abstract

Adaptive Bitrate Streaming (ABR) is a popular technique for providing video media over the Internet. In ABR, the streaming provider splits the video stream into small segments then transcodes them in many different bitrates. So, players can adapt to unstable network parameters minimizing interruptions on playback. However, the computational cost of transcoding a video in many formats can limit its application on live video streaming. Besides, the network overhead of transmitting simultaneously many versions of the same content is a problem. Offloading the transcoding burden to the edge of the network, near the end-users, should alleviate the data traffic burden on the backbone while diluting the computational cost. Users and providers of live video could benefit from a joint scheme that allowed end-user devices to do the transcoding with tolerable latency and delay. We applied Tropos, the agent-oriented software development methodology, to analyze the described scenario and design a multi-agent architecture to deal with the problem of distributed transcoding on Fog-Edge Computing (FEC). The presented architecture consists of three well-defined roles. The transcoder role is intended for those agents on FEC interested in receiving transcoding tasks. The viewer proxy role should be performed by those software agents who will act for the sake of the viewers. The broker role is performed by the agents who will coordinate the tasks for the benefit of the other two. Since FEC is an open environment, distributing transcoding tasks over unknown partners is risky. One of the threats is the risk of untrustworthy partners trying to manipulate the broker by sending it fake information. Literature refers to this kind of manipulation as fake feedback attacks. In this master thesis, we propose combing reward evaluation functions that account for Quality of Service (QoS) with Trust and Reputation Models (TRM) and Multi-armed bandits algorithms (MAB). We present two algorithms, **Re**putation-based **No**de **S**election (ReNoS) and ReNoS-II, designed to online select the best edge nodes to perform the transcoding tasks. We experimented with ReNoS, ReNoS-II, and the other three selecting algorithms in two scenarios to compare them regarding accumulated reward, exploration of available partners, and vulnerability to fake feedback attacks. The outcomes indicate that our proposal can afford rewards gain keeping good QoS as perceived by viewers, besides offering protection against fake feed-

back attacks delivered by untrustworthy transcoders and viewers. Our main contribution is a multi-agent architecture that combines the robustness of T&RM and stochastic MAB algorithms to mitigate the risk of fake feedback attacks, which enabled the employment of unknown partners in open environments. This achievement is in the interest of distributed transcoding applications since it mitigates the risk of employing end-user devices.

Keywords: Multi-agent systems, distributed transcoding, trust and reputation models

Contents

List of Figures

List of Tables

Acronyms

ABR Adaptive Bitrate Streaming.

ACM Association for Computing Machinery.

AIR Antenna Integrated Radios.

CDN Content Delivery Networks.

CGC Complementary Geometric Programming.

CLD Crowdsourced Live Delivery.

CR Certified reputation.

DRL Deep-reinforcement Learning.

DTS Distributed Transcoding System.

FEC Fog-Edge Computing.

FIPA Foundation for Intelligent Physical Agents.

HLS HTTP Live Streaming.

IEEE Institute of Electrical and Electronics Engineers.

IR Interaction Trust.

iStar Intentional Strategic Actor Relationships Modelling.

JADE Java Agent Development Framework.

K-S Kolmogorov-Smirnov.

MAB Multi-armed Bandits.

MAS Multi-agent System.

MEC Mobile Edge Computing.

QoE Quality of Experience.

QoS Quality of Service.

ReNoS Reputation-base Node Selection.

ReNoS-II Reputation-base Node Selection II.

RSU Road-side Units.

RT Role-based Trust.

S-W Shapiro-Wilk.

T&RM Trust and Reputation Models.

UCB Upper-confidence Bounds.

UML Unified Modeling Language.

WAN Wide Area Networks.

WR Witness Reputation.

Chapter 1

Introduction

System designers have faced some real-world applications and systems that are inherently distributed and open. An inherently distributed system is one in which it is impossible to establish a global control or that a simple entity would not pursue full knowledge of the current application status. Open systems are those where participants can leave at any moment, and new participants, often unknown, can enter or be created during normal execution. Such systems are complex and challenging to designers. A way to cope with them is the Multi-agent System (MAS) approach, formed with software agents endowed with autonomy to act on behalf of its owner. A MAS is a society of agents which interact upon specific rules to achieve common or individual goals [10]. Among the advantages attributed to MAS are the following: addressing problems too large to a single entity, providing solutions where expertise is distributed, and providing solutions to inherently distributed problems. Classically, some applications that present those characteristics are decision support systems, networked and distributed control systems, and air traffic control [11].

We argue that providing video over the Internet is one of those complex problems, with technical and human importance. According to [12], one of the reasons for the fast-growing Internet data consumption is the rising demand for video content. The report forecasts that video content will correspond to 80% of all Internet data traffic by 2022. It is also notable that live video streaming is gaining momentum, and the prediction is that live video content over the Internet claims a bandwidth slice from 5% in 2017 to 15% in 2022.

Adaptive Bitrate Streaming (ABR) is a popular technique for providing video media over the Internet. In ABR, the streaming provider splits the video stream into small segments then transcodes them in many different bitrates. So, players can adapt to unstable network parameters minimizing interruptions on playback. However, the computational cost of transcoding a video in many formats can limit its application on live video stream-

ing. Besides, the network overhead of transmitting simultaneously many versions of the same content is a problem. Offloading the transcoding burden to the network edge can deal with these issues.

The appeal of ABR is that experiences in subjective video quality evaluation indicate that Quality of Experience (QoE) is better as higher is the bitrate [13], being much penalized for interruptions [14]. ABR uses HTTP as a transport protocol to facilitate its implementation and adoption [15]. Today, most providers have adopted it due to its advantages. The ABR can be applied to live video distribution but yet should be expected a delay of some seconds. In summary, ABR has become the *de facto* industry pattern among the live video streaming providers. The most used implementation of this technique is the HTTP Live Streaming (HLS) introduced by Apple INC., and then standardized as MPEG-DASH [16].

As said, the computational resources that transcoding demand is not negligible. Indeed, it represents relevant costs for live video providers. Despite pre-stored video transcoding could be postulated, for popular live video providers, to transcode every live channel is resource-consuming. The work of [17] describes two strategies used by Twitch to select the transcoder channel to deal with the trade-off between Quality of Service (QoS) and computational resource consumption.

Many works have tried to transfer ABR functionalities from the cloud to the periphery of the network, seeking out improving the QoS while alleviating providers from the cost of renting expensive cloud servers. We have identified the following motivations for this movement: to reduce the cost of transcoding many video streams at once [3]; to alleviate the data traffic burden off from the network core and reduce latency to end-users [6, 7, 9]; to take advantage of end-user devices' idle resources [4, 5], and profiling end-user preferences to offer personalized QoS [8].

In a cloud-based architecture, the periphery where end-user devices lay is called the Edge or Fog. Taking advantage of those geographically distributed resources is referred to as Fog-Edge Computing (FEC). Indeed, the FEC has plenty of unused computational resources that might be theoretically available at a small network latency. Agents with goals to delegate tasks to nodes in the FEC can use Multi-armed Bandits (MAB) algorithms for selecting the partners. MAB is a simple but powerful machine learning framework for making decisions constrained in time under uncertainty as in the FEC environment [18].

Using FEC devices can be helpful for live video transcoding. However, collaborating with unknown devices can be risky since these devices are neither in control of providers nor viewers. Considering a population where self-interested agents exist, the delegator node is vulnerable to some attacks, such as fake feedback and unfair rating [19]. Untruthful nodes might join others to increase their evaluation artificially while decreasing

others. Maybe due to the probability of the delegates trying to manipulate the delegators, approaches to distributed transcoding are not open to every available node on FEC [6, 7], or does not consider potentially harmful behavior from partners [4, 5]. In this context, a MAS approach must evaluate the risk of cooperation before delegating tasks to partners. In other words, it should know whom to trust. Because of this necessity, Trust and Reputation Models (T&RM) have been frequently considered in the multi-agent design, not only for guiding cooperation but also for partnership, argumentation, negotiation, and recommendation [20].

The work of [21] draws an analogy between reputation systems and MAB policies, comparing some MAB algorithms from the perspective of fake feedback manipulation. The results show that policies are easy to trick. Fortunately, evaluating witness credibility can mitigate the harmful effects of those practices. As presented in [22], applying T&RM is a vital issue when coping with the risk of delegating tasks in open and semi-competitive environments as the FEC.

Thus, in this master's thesis, we use T&RM to cope adequately with fake feedback that should be harmful to MAB algorithms that assume options' payoffs are stochastic. An example of T&RM adequate to deal with the problem is the FIRE model [23]. We also propose to employ separation of responsibilities in a multi-agent architecture to maximize exploration opportunities.

1.1 Problem and Hypothesis

Figure 1.1 illustrates how the concepts presented inter-link to shape our problem domain. The FEC environment is open, dynamic, and has plenty of unused computational resources. ABR depends on transcoding, which is resource-demanding. Offloading the transcoding burden to the FEC nodes seems a good alternative for coping with the problems that live video streaming in ABR faces in the cloud. However, distributed transcoding of live video streaming is geographically distributed, has narrow temporal requirements, and involves many functions and roles. A system to deal with this set of constraints might be complex. Designing such a system would be a more treatable task if approached as a MAS design problem. Since FEC is open, collaborating with possibly self-interested agents might be risky. Since FEC is open, collaborating with possibly self-interested agents might be risky. An example of harmful behavior is when untrustworthy agents form a coalition to manipulate others by providing fake feedback. Fortunately, a MAS can employ T&R models for safer interactions. MAB algorithms are based on a metaphor that relates well to agents' task of exploring and learning about partners and

choosing under uncertainties. However, FEC is a dynamic environment, and agents should watch for not relying on only a partner, even if it is the best partner at the moment.

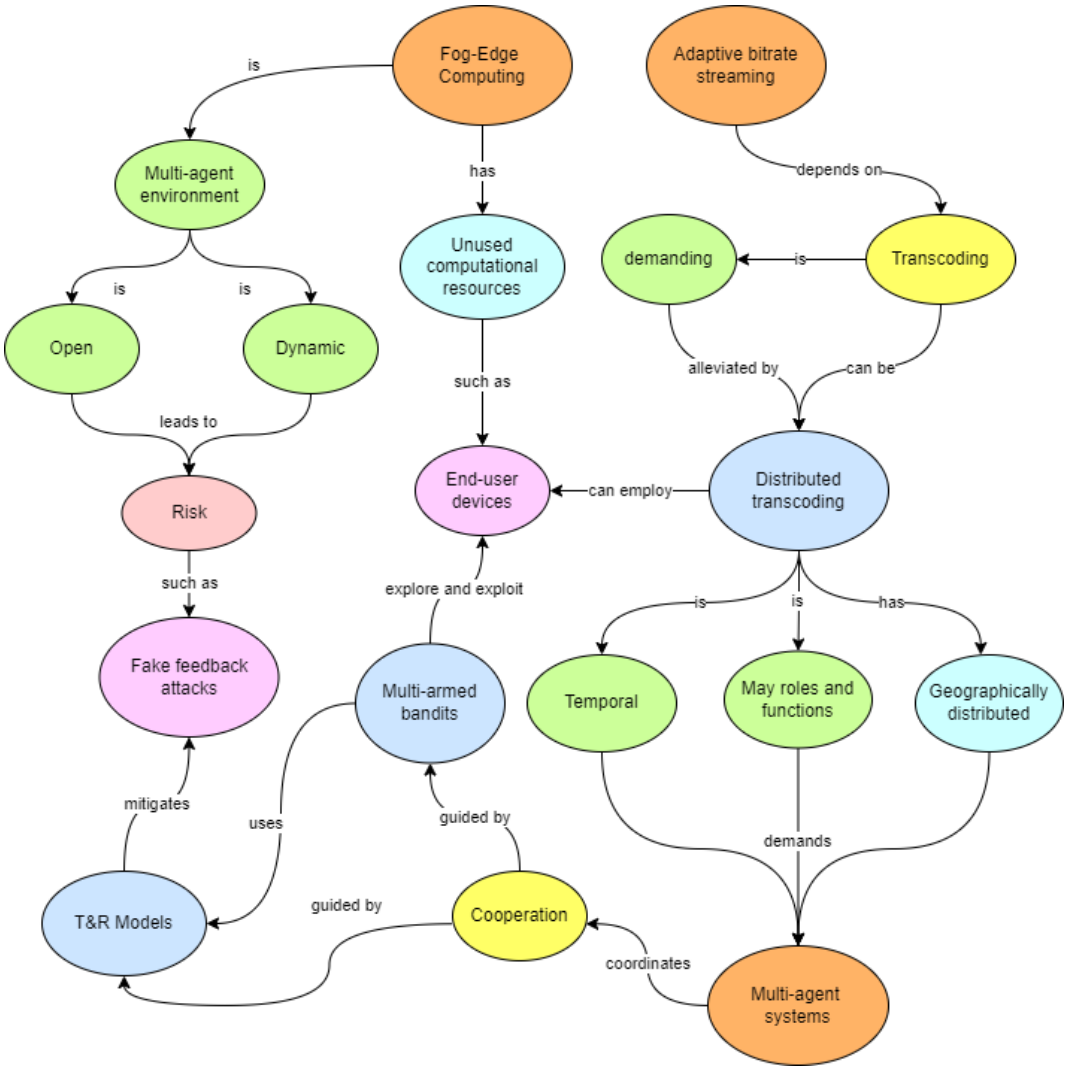


Figure 1.1: The related concepts in this work.

Applying FEC servers worldwide is known to be costly. Viewers engaged in Crowdsourced Live Delivery (CLD) networks frequently have control of devices powerful enough to perform transcoding in the edge [4, 24]. However, approaches to distributed video transcoding on the FEC do not consider the risk of delegating tasks to unknown devices or are not open, which restrains an ample usage of available FEC resources. Considering the conducted literature review, we understand that this still is a good problem that demands further investigation.

We hypothesize that a multi-agent architecture for live video distributed transcoding that applies a T&RM in open environments with unknown partners would improve partners' selection making better use of available resources.

1.2 Objectives

Considering the cited problem and hypothesis, the main objective of this work is to propose a multi-agent architecture that combines the robustness of T&RM and stochastic MAB algorithms to mitigate the risk of fake feedback attacks in open environments. The context of this proposal is the efficient use of computational resources of edge devices for offering real-time video transcoding. We cite as secondary objectives:

- Define the roles, goals, actions, and performance measures for the agents that will compose the architecture.
- Propose an algorithm to combine T&RM and MAB for distributing live-video transcoding tasks to FEC nodes in a balanced way.
- Validate the proposed architecture, indicating its' feasibility for live-video distributed transcoding in open environments.

1.3 Thesis organization

We organized the rest of this document as follows:

- Chapter 2 describes the background concepts upon which we built our proposal;
- Chapter 3 brings selected related work that approached the distributed transcoding problem;
- Chapter 4 details the proposed multi-agent architecture;
- Chapter 5 presents the experiments to validate the architecture, as well the analyses of the outcomes; and
- Chapter 6 presents conclusion with the publications and future work.

Chapter 2

Background

In this chapter, the main concepts related to the problem and the proposed multi-agent architecture are briefly presented.

2.1 Multi-agent overview

For [25], an agent is an entity that can perceive the environment state and act accordingly to modify that state through actuators. Notwithstanding, for [26] an agent is an autonomous entity that functions in the environment inhabited by other agents and processes, where an overview of the environment probably will reveal a multitude of entities. In its turn, [10] defines a MAS as a system compounded of multiple elements capable of autonomous action. Autonomy is the ability to decide by itself what to do to achieve an objective.

A society of autonomous entities brings the probability of a portion of them being self-interested. An agent is self-interested if it selects its actions strategically by the way to obtain the best outcome for itself, do not taking into consideration the necessities of other agents unless it will be beneficial for itself [10]. Commonly, the presence of self-interested agents classify an environment as competitive or semi-competitive, or both, when cooperative and competitive behavior coexist [27]. Nevertheless, being self-interested is different from being malicious, when agents purposely take actions harmful to other agents.

Since agents act in the environment where there are inserted, we have to characterize it during the agent project design. An environment classification based on dichotomies are elicited by [25]:

- Single-agent vs. multi-agent: as seen by [26], single agent environment should be rare in the real world, but it is a useful concept for non-competitive game theory.

- Deterministic vs. stochastic: in deterministic environments, given a state, actions always have only one possible outcome. There is no uncertainty associated with an action result. Otherwise, it is said non-deterministic.
- Static vs. dynamic: an environment is static if it does not change beyond agents acting. If it changes while agents are deliberating, then it is dynamic.
- Discrete vs. continuous: this dichotomy refers to the representation of the environment state. If an agent can always describe the environment with a set of discrete or categorical values, then the environment is discrete. Otherwise, it is continuous. For example, an agent can perceive time as a sequence of discrete episodes or represent it as the number of milliseconds since the system is running.
- Episodic vs. non-episodic: episodic environments are those in which interaction is divided into atomic episodes, and what an agent does in one episode does not influence agent performance in other ones. Otherwise, agent performance is linked to past episodes that is why they are sometimes related as historical.
- Fully Observable vs. partially observable: if the agent can access the whole set of variables that describe the environment state, then the environment is fully observable. However, if variables are sometimes not accessible or information is outdated, then the environment is partially observable.

Additionally, we have to distinguish between open and closed distributed systems. In this work, we consider that a multi-agent environment is open if the agents act on behalf of distinct owners and come from different sources. A popular open multi-agent environment is the Internet.

A rational agent is an agent that evaluates the possible actions and then selects the one that results in the best outcomes. Two important processes for rational agents are deliberation and means-end reasoning. Deliberation refers to the process of choosing what goals an agent wants to achieve, while means-end reasoning is the process of figuring out how to achieve those selected goals [28]. Considering non-deterministic environments, the agent chooses the ones that result in the best-expected outcome. The manner agents embody rationality, among others, is classified by [10] as reactive, symbolic reasoning, or hybrid, as follows:

- Rational agents can use symbolic logic to decide what to do. Symbolic means that perceptions must be transducted into symbols, which are predicates of the agent's logical system. Those predicates will give form to the agent's beliefs database. Based on its database and a set of transformation rules, the agent can reason about what action to do. However, there are some disadvantages. The agent's logical system,

despite the sound and complete properties, might not be decidable. If decidable, calculations may take so long that the environment can change, and hence the agent selected an inappropriate action based upon deprecated information.

- Reactive agents are those who directly map actions to perceptions without explicitly reasoning about probability distributions over states, characterized by a tight sensor-action loop. A manner to implement reactive agents is using a look-up table. Its decision cycle is often on the order of milliseconds or faster, but a complex environment should require a long list of associations on the table.
- Hybrid agents combine symbolic reasoning and reactive attempting to benefit from both. A usual approach is to set up reactive rules linked to survival behavior in higher priority and symbolic reasoning for performance-related behaviors.

2.1.1 Types of interaction

Agents must communicate if they should solve problems in a distributed manner. Interaction protocols organize agent communication, so messages exchange be efficient and coherent without violating agent autonomy. According to [28], some relevant aspects that interaction protocols should consider, among others, are: determining shared goals and common tasks, avoiding unnecessary conflicts, and pooling knowledge and evidence.

In [29], types of interaction are classified accordingly with the goals, resources, skills, and situations. The introduction elicits the following classes of interaction: communication, coordination, planning, collaboration, cooperation, competition, and negotiation. Communication is when agents merely pass messages to each other. When the objective of message exchanges is managing the inter-dependency among tasks, then the interaction is coordination. When the goal is arranging actions into sequences, the type of interaction is planning. Collaboration is when agents communicate to enable each other to achieve their individual goals. When agents communicate to achieve a common goal, the interaction type is cooperation. If agents are antagonists and compete for resources, the interaction is competition. Finally, when they communicate to reach agreements on matters of mutual interest, then the interaction type is negotiation. As we can see, this classification clarifies that many types of interaction can occur among the same group of agents. For example, a planning interaction might lead to consequent negotiation or competition.

According to [28], in environments with a multitude of agents, actions needed to be coordinated. Coordination is necessary since no agent has sufficient competence, resources, or information to achieve the global objectives of the system. The decomposition and distribution of tasks is a basic approach to coordinating a group of agents. Smaller tasks require less powerful agents and consume fewer resources. Task decomposition can be

inherent to the system, guaranteed by design, or agents can perform the decomposition executing a plan of actions. According to the author, once tasks are decomposed, the distribution has to consider the following constraints:

- Do not overload critical resources.
- Tasks must be assigned to agents with matching capabilities.
- Only an agent with a wide view should assign tasks to other agents.
- Assign overlapping responsibilities to agents to achieve coherence.
- Highly interdependent tasks should be assigned to agents in spatial or semantic proximity.
- Reassign tasks if necessary for completing urgent tasks.

In our proposal, interaction occurs in two distinct moments to each pair of agents. An overseeing agent selects candidate partners that are adequate to receive a task; they negotiate the conditions under which the work should be done and, then, after agreeing, they cooperate to achieve the common goal of transcoding a live streaming event.

2.1.2 Tropos methodology

Tropos is a software development methodology for agent-oriented programming, which allows the explicit representation of mentalistic notions as goals and qualities. The two aspects that differentiate it from other software development methodologies are the early requirements analysis and the agent-centric nature, which are present in all software development phases [30]. The Tropos methodology guides the development of agent-oriented solutions through a set of phases, through which the developer gradually refines the representation of the system. In the initial phases, Tropos represents the agent model in diagrams, as the one illustrated in Figure 2.1. Figure 2.2 presents some of the visual components used in Figure 2.1.

The visual language is based on i* framework[1]. The diagrams' visual components represent actors, agents, roles, objectives, tasks, plans, and resources [32]. This work uses Tropos concepts that we must define.

For the Tropos methodology, an Actor is an entity that pursuit intentionality. The actor is a general concept from which roles and agents are specialized. Diagrams illustrate actors by circles. By its turn, an agent is a physical manifestation of an actor, i.e., an agent is an actor identified by a proper name. The agent graphical representation is a

[1]Intentional Strategic Actor Relationships Modelling (iStar) [31].

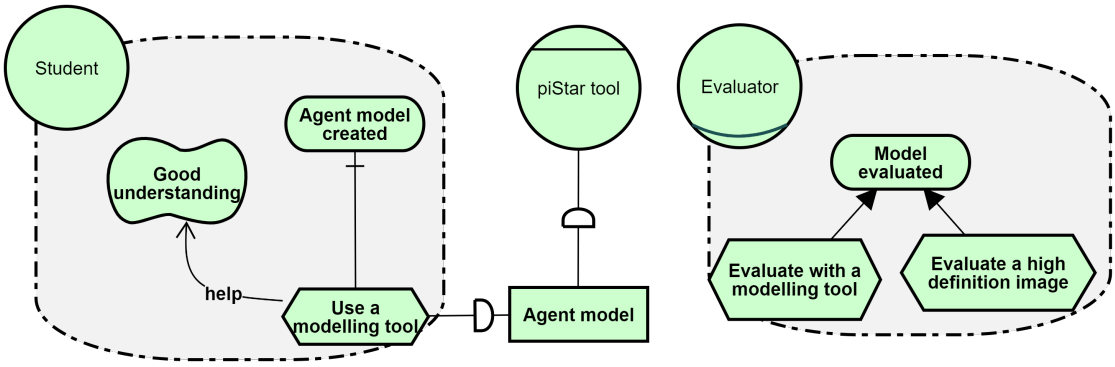


Figure 2.1: Example of a Tropos diagram.

circle with a straight line in its superior basis, indicating an intelligent agent. A role is an abstraction that encapsulates some behaviors and intentions that actors can fulfill. In diagrams, roles are circles with a curved line in its inferior part, representing an actor with some kind of specialization in the environment in which it is included.

We also have to differentiate between hard-goals and soft-goals. A hard-goal is an actor's strategic goal with clear satisfiability criteria. A soft-goal is related to agents' welfare expectations, which means how an actor desires some interest should be complete. In the diagrams, the ellipses graphically represent the hard-goals, while elements similar to clouds represent the soft-goals.

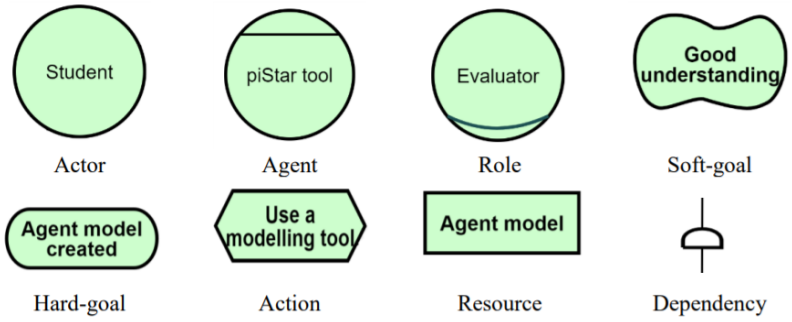


Figure 2.2: Components of a tropos diagram.

A capability is the ability to choose and execute a plan to fulfill a goal, constrained by the environment status and the occurrence of specific events [30]. In diagrams, capabilities, plans, and individual tasks are represented by the visual component task. The resource represents a physical or informational entity that an actor or agent needs to execute a task. Resources can be generated or used by the components involved in the tasks that enable the agent to achieve its goals. The diagrams present tasks as hexagons and resources as rectangles.

Generally, these elements relate by themselves through inter-connections. Each connection represents the relationship level between two visual components, as presented in Figure 2.1. When a AND association links two elements to a goal, the agent must execute both before achieving the goal. Distinctly, when this is an OR type association, executing any of the elements is enough to achieve the goal. A qualification serves to connect the soft-goals to the elements that they qualify, like in Figure 2.1 where a qualification says that good understanding *helps* the actor Student to use a modeling tool. Graphically, a T-form arrowhead indicates the refinement AND. A solid arrowhead directed to the origin represents a refinement OR. A traced line represents a qualification [31].

As presented in [30], the Tropos methodology specifies five main development phases to encompass the whole software design process:

- In the early requirement phase, the system as-is is analyzed and then decomposed into social actors along with their intentions and relations.
- In the late requirements phase, a new actor is introduced to represent the system to be modeled.
- The objective of the architectural design phase is to produce the system specification in terms of a detailed architecture of interconnected subsystems and data repositories. In this phase, the system introduced in the late requirements phase is divided into new actors representing the software agent subsystems.
- The detailed design defines the plans of actions that each agent should be able to execute to achieve their goals. The objective is to list and specify the agents' capabilities.
- The implementation phase is when the software agents are constructed in a specific computational platform.

2.2 Trust and reputation models

To face the uncertainty involved in collaboration throughout FEC, a rational agent A, before delegating a task, which its welfare depends on, to an agent B, must somehow compute the probability of B completing the task successfully. This probability means how much agent A trusts B relative to the delegated task's completion. Nevertheless, an agent may not be able to evaluate the trustworthiness of a counterpart if they have never met. A recurrent solution to the lack of direct interaction is to ask others about the opinion they have of the possible partner concerning the coted task. A weighting of

the trust that a node receives from a collective is called reputation. These definitions of T&R are in line with the ideas presented in [33].

The base of reputation calculations is averaging the opinion of third parties that had directly interacted with someone. However, for many factors, those witness reports might be inaccurate. According to [34], agents tend to be self-interested inside an open environment. Because of this, many T&RM have mechanisms to evaluate witnesses' credibility and use this evaluation to weigh the received reports.

Bringing up the distributed transcoding problem domain, an agent that frequently fails to complete the delegated tasks, and does not contribute to improving the delegator's perceived QoS should receive a low trust value. Since viewers of live video streaming could obtain the stream directly from the stream provider at the cloud, there is no sense to select a poor transcoder in the FEC. In this scenario, we should adopt the decision-making process proposed by [35]. Hence, the delegator should remove the nodes not capable of keeping themselves above a trust threshold from the pool of possible partners. The same must occur when witness credibility decreases below a credibility threshold.

There are many models of interaction based on T&R in the literature, like Marsh [35], REGRET [36], Sporas [37], and FIRE [23]. One of the aspects where those models differ is information sources. All consider the direct and indirect interaction on its calculations, but FIRE introduces certified reputation. Gathering reputation from witnesses implies communication costs since agents must explore the social graph to acquaint the existing evaluations. If available, an evaluation rating certified by a trusted authority can be acquired by only one interaction.

Authors in [38, 39, 40] compared T&RM and proposed a meta-model that allows reasoning about the parameters of the models at run time. Considering the heterogeneity of the devices in the FEC, a rational agent could reason at the meta-level to adjust requirements for limited resources like storage, network, and energy. But these aspects are left as future work in this research.

2.2.1 Marsh

The Marsh model defined in [35] is a numerical T&RM that uses only the concept of direct trust, i.e., it considers only interactions between the trustor and the trustee. The Marsh model defines the concept of situational trust, which represents the willingness that an agent has to delegate a task to another agent, the trustee, regarding a specific situation. Its main objective is to aid the trustor to decide delegate or not the achievement of one of its goals to another agent based on past interactions. In the Mash model, calculations take Equation 2.1.

$$T_x(y,\alpha)^t = U_x(\alpha)^t \cdot I_x(\alpha)^t \cdot \widehat{T_x(y)^t} \tag{2.1}$$

For explaining Equation 2.1, let us say that exists a specific situation α that interests the agent. The value $U_x(\alpha)^t$ represents the utility that the agent will receive if the situation α is achieved. The value $I_x(\alpha)^t$ is the importance of the situation α to the wellbeing of the agent. The value $T_x(y)^t$ represents the result of the past interactions with the agent y. The author in [35] suggests that the agent can approach the result of past interactions in three moods, which are optimistic, pessimistic, or realistic. If the agent is pessimistic, it will consider only the worst results. If the agent is optimistic, it will consider only the best result. Otherwise, the realistic agent will consider the averaging of all past results. The trust value obtained by Equation 2.1 is relative to a specific moment in time, which is the meaning of the modifier t that accompanies all factors in the equation.

With the trust value obtained from Equation 2.1 in hand, the agent will confront it with a cooperation threshold. The cooperation threshold must consider the risk that represents the non-achievement of situation α to the agent and the cost of cooperating with others. The agent will only delegate the task of achieving α to another agent if the trust in that agent surpasses the cooperation threshold.

Since we assume that collusion between self-interested agents is a possibility in open environments, we do not recommend using the Marsh model in this situation. For supporting this, we argue that considering just direct interaction does not protect the trustor when partners decide to manipulate it.

2.2.2 Sporas

The Sporas presented in [37] is a model designed for sparsely linked societies. Agents in society send their ratings to a central evaluator agent that aggregates them in reputation values. The model does not predict direct interactions between the evaluator and the other agents. Thus, it does not produce trust values, only reputation values. Another limitation is that an agent only interacts with one agent per time. If two agents interacted more than once in their lives, the evaluator only considers the last sent rating. This characteristic can be limiting depending on how an application deals with the time dimension. For example, probably, a streaming provider is frequently interacting with many users at once. In this scenario, it should be hard to manipulate the time dimension in a way that interaction always involves only two agents.

The evaluator calculates the reputation values using the recursive Equation 2.2. The notation R_n is the reputation calculated in the n^{th} interaction. The Θ is the number of interactions considered. The function Φ is a decaying function presented in Equation 2.4.

The R_i^{other} factor is the reputation of the agent that sent the rating W_i. The reputation value range is between zero and D ($[0, D]$). Authors in [37] suggest a D value of 3.000, although rating range is between 0.1 (terrible) and 1 (perfect). Thus, Equation 2.3 is necessary due to the range adjustment between reputation and rating values. Initial reputation value is zero, so an agent that switches its identity does not gain anything.

$$R_i = R_{i-1} + \frac{1}{\Theta} \cdot \Phi(R_{i-1}) \cdot R_i^{other} \cdot (W_i - E_i) \tag{2.2}$$

$$E_i = R_{i-1}/D \tag{2.3}$$

The parameter θ is the acceleration factor of the decaying function presented in Equation 2.4. As smaller is the adopted value for θ, more abrupt is the decaying of Φ.

$$\Phi(R_{i-1}) = 1 - \frac{1}{1 + e^{-\frac{R_{i-1}-D}{\theta}}} \tag{2.4}$$

The Sporas model also provides a reliability value that measures how confident is the evaluator about the produced reputation values. It is called the Reliability Deviation (RD). The RD value is given by Equation 2.5, which gives us the minimum of the quadratic difference using recursion, as Equation 2.2. In Equation 2.5, RD_n^2 is the reliability deviation value in the n^{th} interaction, λ is the forgetting factor (range in $[0, 1]$), and T_O is the number of effective observations.

$$RD_i^2 = \lfloor \lambda \cdot RD_{i-1}^2 + (R_i^{other}(W_i - E_i))^2 \rfloor / T_0 \tag{2.5}$$

2.2.3 FIRE

The FIRE T&R model presented in [23] can use several sources of information to calculate the trustworthiness and reputation of a party. The FIRE model use four components for trust calculation: Interaction Trust (IR), Witness Reputation (WR), Role-based Trust (RT), and Certified reputation (CR). IR resulted from direct interaction, also known as direct trust. WR is a weighting of trust evaluations provided by third parties. In its turn, RT is the trust based on role-based relationships between agents, e.g., agents owned by the same proprietary. CR is evaluations made by an authority and provided by target agents as proof of past good performance. Among those presented in this work, the FIRE model is the more adequate. Differently of Marsh model, it can use many information sources, not only direct interaction. It also allows agents to interact with more than one partner at once, which is an advantage when comparing it with the Sporas model.

In the FIRE model, trust evaluations are calculated by Equation 2.6, where:

- k is one of the components for trust calculation;
- $T_k(a, b, c)$ is the trust that agent a has of agent b in relation to subject c;
- $R_k(a, b, c)$ is the set of all ratings collected;
- $\omega_k(r_i)$ is the relative weight of rating r_i and v_i is the value of the rating.

$$T_k(a, b, c) = \frac{\sum_{r_i \in R_k(a,b,c)} \omega_k(r_i) * v_i}{\sum_{r_i \in R_k(a,b,c)} \omega_k(r_i)} \tag{2.6}$$

A different weighting function is defined for every information source. The weighting function for IR is defined in Equation 2.7 and represents the degradation of the rating reliability over time.

$$\omega_{IR}(r_i) = e^{-\frac{\Delta t}{\lambda}} \tag{2.7}$$

Before calculating the weighting function of the WR component, witnesses' credibility must be evaluated. It was proposed in [34] to compare witness' ratings with those directly obtained, and then the deltas were applied to the FIRE model to derive a credibility factor. Assuming that an agent a wants to evaluate the credibility of a witness w and IR ratings are available, the Equation 2.8 is applied, where v_k is the value by the witness and v_a by direct interaction. If the difference between the values is higher than an inaccuracy threshold i, the witness is penalized with the lowest possible value (-1).

$$v_w = \begin{cases} 1 - |v_k - v_a|, & \text{if } |v_k - v_a| < i \\ -1, & \text{if } |v_k - v_a| \geq i \end{cases} \tag{2.8}$$

The second step is to apply Equation 2.6 as if calculating the IR component. If direct interaction values were not available, a default credibility rate is assigned to every witness. Equation 2.9 demonstrates this reasoning, where T_{DWC} is the default credibility value.

$$T_{wc}(a, w) = \begin{cases} T_{IR}(a, w, c_{WC}), & \text{if } R_{IR}(a, w, c_{WC}) \neq 0 \\ T_{DWC}, & \text{otherwise} \end{cases} \tag{2.9}$$

Finally, Equation 2.10 is the weighting function for the WR component. It takes weights calculated with Equation 2.7 since rating reliability decays with time, but it is multiplied by the credibility factor. If T_{wc} for a witness is less than zero, it means that the witness is not trustful at all, so its reports should not be taken into account.

$$\omega_{WR}(r_i) = \begin{cases} 0, & \text{if } T_{wc}(a,w) \leq 0 \\ T_{wc}(a,w) * \omega_{IR}(r_i), & \text{otherwise} \end{cases} \tag{2.10}$$

Untruthful delegate nodes might plot with each other to artificially increase its evaluation and decrease the others. As advocated in [34], evaluating witness credibility can mitigate the harmful effects of those practices, or, at least, it would take more time until a delegator node was tricked into trusting an untrustworthy partner.

Another measure that the FIRE model provides which is of our interest is the reliability of the reputation value. While trust and reputation can represent the expected performance of an agent in a specific task, the reliability value tells how likely it is that the agent delivers as expected. Using the FIRE model, we can obtain the reliability of our reputation values by executing two steps. The first step is calculating the rating reliability with Equation 2.11, the second one is calculating the deviation reliability with Equation 2.12.

Rating reliability calculated with Equation 2.11 represents how reliable is the set of ratings R_k. Since the weighting function represents the reliability of one rating considering time decay, the sum of all weightings gives us an idea of the reliability of the set, considering the set length and the recency of the elements in the set. Intuitively, as greater is the sum of the weights, as closer one gets the rating reliability. The factor γ_k is an adjustment for the curve slope since every component has a different weighting function.

$$\rho_{R_K}(a,b,c) = 1 - e^{-\gamma_k \cdot (\sum_{r_i \in R_k(a,b,c)} \omega_k(r_i))} \tag{2.11}$$

Equation 2.12 measures the dispersal of the rating set around the trust value T_k with the relevance adjusted to each correspondent weight. The idea is that as larger is the deviation from the trust values, as volatile is the agent, thus, less reliable it is.

$$\rho_{D_K}(a,b,c) = 1 - \frac{1}{2} \cdot \frac{\sum_{r_i \in R_k(a,b,c)} \omega_k(r_i) \cdot |v_i - T_k(a,b,c)|}{\sum_{r_i \in R_k(a,b,c)} \omega_k(r_i)} \tag{2.12}$$

Finally, the reliability of the trust value T_k concerning the component k is the combination of both the rating and deviation reliability given by Equation 2.13.

$$\rho_K(a,b,c) = \rho_{R_K}(a,b,c) \cdot \rho_{D_K}(a,b,c) \tag{2.13}$$

An overall trust evaluation is calculated using Equation 2.14, which is a combination of all components of the FIRE model. In that, k varies in the set of all existing components ($C = \{IR, WR, RT, CR\}$), and ω_k is the combination of the component coeficient W_k

with its reliabity ($\omega_k(a,b,c) = W_k \cdot \rho_k$(a,b,c)). Likewise, FIRE model provides the overall reliability of $T(a,b,c)$ by applying Equation 2.15.

$$T(a,b,c) = \frac{\sum_{k \in C} \omega_k(a,b,c) \cdot T_k(a,b,c)}{\sum_{k \in C} \omega_k(a,b,c)} \tag{2.14}$$

$$\rho_T(a,b,c) = \frac{\sum_{k \in C} \omega_k(a,b,c)}{\sum_{k \in C} W_k} \tag{2.15}$$

2.3 Fake feedback

In this work, fake feedback is defined as the deliberate act of providing unrealistic feedback ratings about an entity or service to misguide others' evaluations and manipulate their choices. We distinguish between fake feedback and inaccurate report. There would be many reasons for a witness providing inaccurate reports, like environmental factors, temporary or not, or even differences in the agents' mental state, like unmatching evaluation scales. At this point, we focused on harming a trust model or system intentionally.

The work on [41] presents a historical about fake feedback attack generations to the trust system of the Chinese e-commerce platform Taobao. In that work, we noted the importance of user plotting to achieve success in the task of manipulating sellers' ratings. In that case, a T&R model was proposed to deal with the fake feedback problem in the platform. The work indicates that fake feedback is a problem for many online, open, and dynamic environments.

Following [19], T&RM are essential to collaboration where parts may not have prior knowledge about each other. However, T&RM are not free from manipulative attacks. The introduction in that work lists some kinds of attacks from which T&RM are vulnerable, like discrimination, self-promoting, white-washing, and fake feedback, among others. Fortunately, there are strategies that T&RM can apply to mitigate the risk these threats bring to users. One example of such strategies is evaluating the witnesses' credibility before accepting the feedback [42, 43, 23]. In the works of [44] and [45], authors deal with the problem of fake feedback attacks to the cloud environment proposing domain-specific T&RM.

According to the cited works, the fake feedback problem would be mitigated in systems, platforms, online, open and dynamic environments with T&RM strategies. Thus, in this research, we adopt this understanding.

2.4 Multi-armed bandits algorithms

MAB algorithms are applied when an agent has to choose among uncertain alternatives. Agents do not know the options reward distribution but have some confidence that there are choices better than others. The uniform exploration phase tries randomly for a certain number of times to estimate the reward distribution of every tried option [18]. Afterward, agents can use the built knowledge to choose the better option among the tried ones in the exploitation phase. Exploitation does not mean that the agent undoubtedly identified the better option since the actual reward distributions are unknown and might be an optimal choice different from the one taken.

The difference between the option taken and the optimal one is called regret. With limited time to play, there is the necessity to balance between the exploration and the exploitation phases to maximize the cumulative reward or minimize the regret. According to [18], $\mu(a)$ is the mean of the rewards obtained from the arm $a \in A$, A the set of all available arms. Exists an a^* for which $\mu(a^*) \geq max_{a \in A}(\mu(a))$, and for simplicity, we call $\mu(a^*)$ μ^*. Considering that there were T rounds, and a_t is the option selected at the round $t \geq T$, the regret at the round T is defined by Equation 2.16.

$$R(T) = \mu^* \cdot T - \sum_{t=1}^{T} \mu(a_t) \tag{2.16}$$

Many algorithms implement different strategies to balance exploration and exploitation phases. To exemplify, we can cite: ϵ-Greedy, ϵ-First, ϵ-Decreasing, and the Upper-confidence Bounds (UCB) family.

2.4.1 ϵ-Greedy

The ϵ-Greedy algorithm uses a constant ϵ in the range [0,1], which is the probability of using exploration. At each round, there is a probability of ϵ that the player will choose an option by chance, and a probability of $1 - \epsilon$ that it will make the greedy decision, i.e., choose the option with the highest average reward [46]. Choosing an adequate value for ϵ is left to the user. Algorithm 1 describes the pseudocode for ϵ-Greedy.

Algorithm 1 ϵ-Greedy algorithm

```
1: procedure ε-GREEDY(ε, arms)
2:     p ← Random()
3:     if ε > p then
4:         Select random([arm ∈ arms])
5:     else
6:         Select argmax([reward(arm) ∈ arms])
```

2.4.2 ϵ-First

The ϵ-First is similar to ϵ-Greedy with a constant ϵ to determine the exploration round proportion, but the whole exploration is taken in the beginning to estimate the reward distribution first. After that, the agent will always choose the option with the highest reward estimated. It assumes there is a fixed number of rounds available, a maximum number of turns, which is referred to as the Horizon (H) [46]. The number of exploration plays at the beginning is $H \cdot \epsilon$, and the number of exploitation plays at the ending is $H \cdot (1 - \epsilon)$. When the maximum number of plays is unknown, the user should repeat the phases to keep the proportion between phases coherent with the chosen ϵ. In those cases, the user should pick a discretionary H, and then split the run into episodes of H plays each. Algorithm 2 presents the pseudocode for ϵ-First.

Algorithm 2 ϵ-First algorithm

```
1: procedure ε-FIRST(ε, arms, round, H)
2:     if mod(round, H) < (ε · H) then
3:         Select random([arm ∈ arms])
4:     else
5:         Select argmax([reward(arm) ∈ arms])
```

2.4.3 ϵ-Decreasing

The ϵ-Decreasing is similar to ϵ-Greedy, as it uses an ϵ which represents the probability of exploring in each round. The difference is since the algorithm starts with a value for ϵ and, as the run is played, the ϵ value is decreasing to the point when choosing the highest reward options is very likely [46]. Thus, besides having to select an appropriate value for ϵ, the user should select a function upon which the ϵ value will decrease. Authors in [46] point that this algorithm belongs to the zero-regret class as the expected regret tends to zero as time goes by. Algorithm 3 shows the pseudocode for ϵ-Decreasing, where parameter $function$ must be a decreasing function which domain is $[0, 1]$.

Algorithm 3 ϵ-Decreasing algorithm

```
1: procedure ε-DECREASING(ε, arms, round, function)
2:     p ← Random()
3:     if (ε · function(round)) > p then
4:         Select random([arm ∈ arms])
5:     else
6:         Select argmax([reward(arm) ∈ arms])
```

2.4.4 UCB family and UCB1

In the UCB family of adaptive bandits algorithms, they approach the uncertainty with the optimistic assumption that options are as good as we could evaluate until the moment [18]. In the simple UCB1, every arm is selected by the player at least once, and then it will pick out the one with the highest observed reward [47]. Equation 2.17 is the UCB1 algorithm core, where A is the available options set, μ is the average of option reward, T is the number of rounds up the moment, and n_t represents how many times an option is selected. As much as an option is selected, more unlikely it is to be selected unless its reward was higher than others.

$$UCB1(a) = \arg\max_{a \in A} \left(\mu(a) + \sqrt{\frac{\alpha \log T}{n_t(a)}} \right) \tag{2.17}$$

2.4.5 Other MAB policy classes

The problem of contextual bandits, also known as bandits with aside information, can be described as follows: on round 1, the context, an *n-Tuple* value, is announced. The player choices the arm a_1 and observes the obtained reward. After the context is announced in subsequent rounds, the player tries to preview the best arm to pull based on the context and the accumulated information from the previous rounds [48]. The game, then, continues following the same structure.

On the space problem known as the adversarial MAB, a new player called the adversary is introduced. On each round, as the player selects the arm, simultaneously the adversary chooses the reward [49]. The aim is to eliminate any reward distribution assumptions making the solutions more generic without leaving some guarantees aside.

2.4.6 MAB and T&RM

This work addresses a problem subtly different from contextual bandits and adversarial MAB. The player is not sure about the rewards reported by the witness but only the pulls made by himself. However, witnesses' reports can be interpreted as contextual information so that our problem might be in the contextual bandits' problem class. But we propose to look after the mean represented by $\mu(a)$, regardless of the ideas in the algorithms.

Authors of [50] justify their approach by setting a scenario for MAB where arms' rewards are stochastic but might be corrupted by an adversary. When such a situation occurs, rewards' feedback will be partially stochastic and partially not. Thus, stochastic bandits algorithms do not apply since they are easily manipulable by the information changed by the adversary. In this case, the adversarial bandits will deliver a poor performance since they cannot take advantage of stochastic assumptions. Thus, the authors

formalize a model and propose an algorithm that ensemble to active arm elimination algorithm to solve the problem. Despite fake feedback attacks being an instance of the corrupted reward scenario traced by [50], this work proposes to treat the information in a manner that stochastic algorithms can still be applied.

An analogy between reputation systems and MAB policies is drawn by [21]. Authors point out that reputation systems and MAB algorithms have the same objective of solving selection problems. Authors argue that MAB policies could improve reputation systems turning these systems more resistant to manipulation. We claim that T&R evaluation is separated from algorithms for selection since the objectives of T&RM are broader than just selecting among possible partners, even that this objective can be related to the trust delegation defined by [33]. We can say that an agent who employs the T&RM does not necessarily use reputation for selecting partners. Besides, some T&RM deal with untrustworthy agents in their proper manner. On the other hand, MAB algorithms are employed in scenarios where reward corruption is possible, yet trust is not usually considered, like in [50].

2.5 Fog-edge computing environment

In [1], the authors defined FEC as a complement of cloud computing that employs devices on the edge of the network to improve the quality of service towards a service continuum. Figure 2.3 illustrates the FEC layer architecture. The authors divide the FEC into three layers: Inner-edge, middle-edge, and outer-edge.

The inner-edge layer corresponds to networks where the covered area is as large as a country. In the inner edge are placed the infrastructure for geo-distributed cache and the processing centers of Wide Area Networks (WAN), as the Content Delivery Networks (CDN) mentioned in [2]. The objective of the inner-edge layer is to improve QoE lowering the network latency. The middle-edge layer corresponds to the environment where are set-up MANs, LANs, Wireless LANs, and the cellular network. The middle-edge is the common understanding of the fog computing layer. The outer edge is also known as the far-edge and things layer where we find users' devices like mobiles, sensors, and actuators.

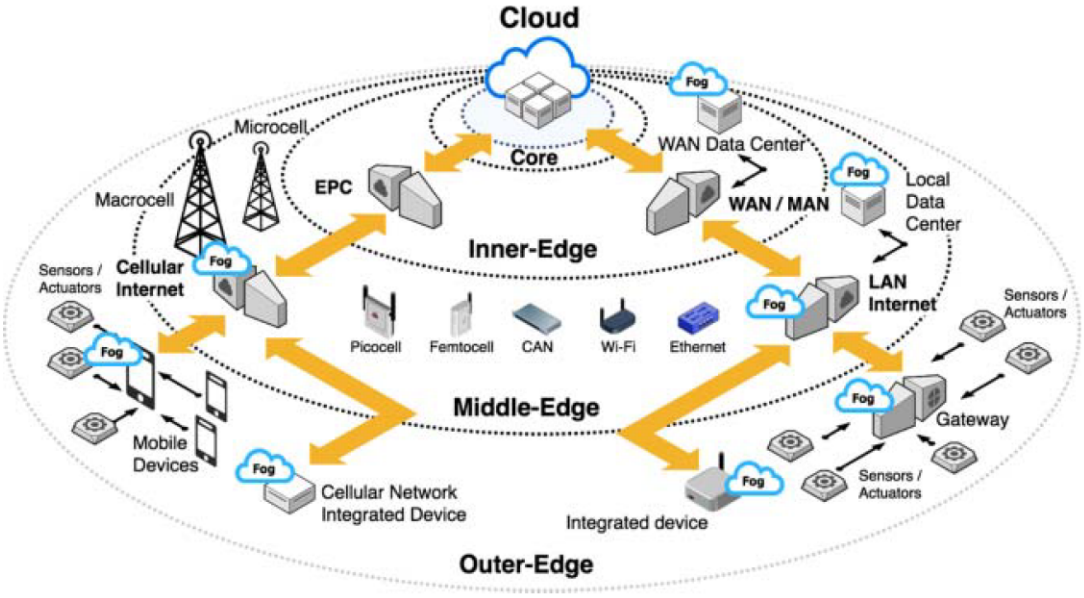


Figure 2.3: The FEC layer architecture [1].

As a complement to the Internet infrastructure, the FEC inherits some of its characteristics. As some FEC nodes have the autonomy to make their decisions to improve performance, we can describe them as agents. Thus, we are interested in defining what type of agent environment the FEC is. Adhering to the classification presented in [25] and reported in this work in Section 2.1, we can list the FEC properties as follows:

- Multi-agent - it is expected that agents in the FEC could cooperate to improve overall performance;
- Non-deterministic - agents would not be able to determine the next state only by evaluating the effects of their actions;
- Dynamic - assuming that nodes can join and leave the network unexpectedly;
- Continuous - some environmental measures can be discretized but not all of them, and not for all the purposes;
- Sequential - as long as decisions made in the present will influence performance in the future;
- Partially observable - information about the environment and other agents would remain outdated most of the time.

2.6 Adaptive bitrate streaming

The central idea of ABR is transcoding the same video input into segments of different bitrates. Figure 2.4 presents a typical workflow to deliver a live video event using ABR including five steps [2]: i) the video content is obtained and pushed up to a server; ii) the video stream is decoded and then re-coded to a convenient format for transmission; iii) the video stream is split into segments and published on HTTP servers along with manifest files; iv) Regionally distributed CDN nodes minimize latency; v) viewer's player performs buffering before decode and play the stream.

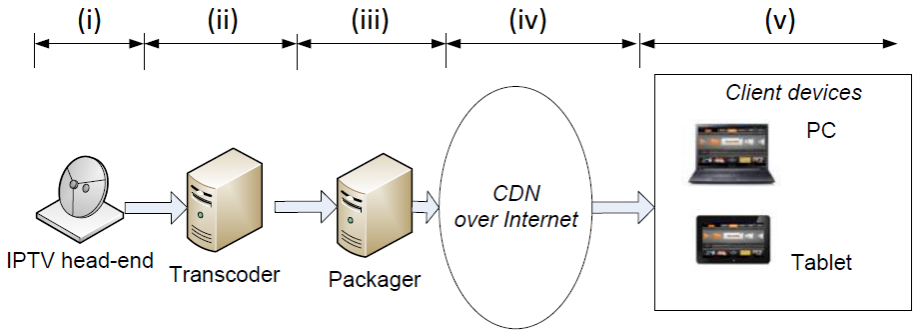


Figure 2.4: ABR components in live video broadcast. Adapted from [2].

The buffering technique employed by viewers' players is the key characteristic in determining the predictability of video segment requests. There exist many algorithms that different players should apply. Following the work in [51], there are three categories where those algorithms can fit in: throughput-based algorithms, buffer-based adaptation, and time-based adaptation.

Throughput-based algorithms take constant measures of network bandwidth to evaluate TCP throughput and then schedule video segment requests based on the current state of a buffer. The Buffer-based adaptation class observes the buffer occupancy itself to determine when and under what bitrate representation each video segment should be requested. Finally, in Time-based adaptation, downloading time is considered rather than TCP throughput and then uses a pre-computed buffer-map to select the appropriate video representation.

As commented, the audience of online videos do prefer high bitrate versions, but interruptions harm QoE [13, 52, 14]. The work in [14] defines the utility of an online video session as the combination of the average bitrate counterbalanced by the playback smoothness. Playback smoothness is a ratio between time spent rebuffering and the total time of video exhibition. Equation 2.18 is the Joint Utility Function (JUF) defined in [14, p. 4], where $\overline{v}_N$ is the playback utility, $\overline{s}_N$ is the playback smoothness, and $\gamma > 0$ is a parameter to prioritize playback utility with playback smoothness.

$$JUF = \overline{v}_N + \gamma \cdot \overline{s}_N \tag{2.18}$$

Transcoding on distributed environments

The computational demand generated by transcoding in ABR led to the search for alternatives that could alleviate stream providers from part of the costs. A natural approach is employing cloud computing processing, which is less costly than maintaining robust private servers [53]. Netflix has been using cloud computing to make transcoding [54]. However, transcoding in the core results in relatively high latency and consequent high delay in live video streaming. Edge-servers can offer a lower delay than those in the core on the trade of major costs. Another proposal is to use peer-to-peer sourced transcoding [55], but a disadvantage of this method is the low reliability of end-user devices. Some others, like [4, 5, 6, 8, 9], propose a combined architecture where nodes in the core, middle and outer edge could be employed in transcoding. Figure 2.5 is a representation of how could be a combination of transcoding on core, middle-edge, and outer edge. The original video content is acquired and transcoded to a high bitrate representation, then distributed through the network. When required, edge devices transcode the stream to lower bitrates, saving bandwidth at the backbone and reducing latency.

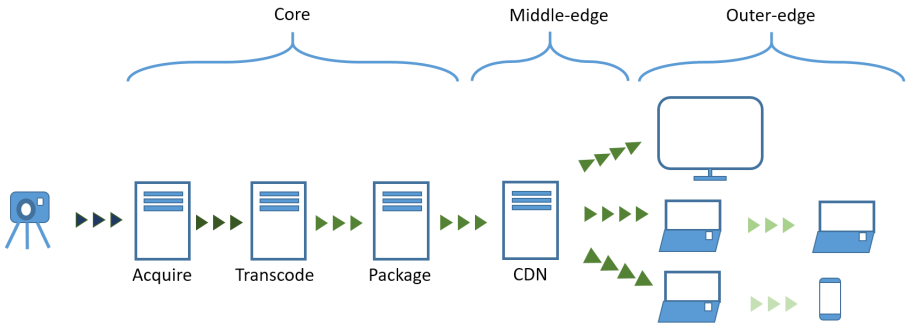


Figure 2.5: Distributed transcoding through FEC layers.

2.7 Statistical evaluation

In Chapter 5, we present a considerable amount of data delivered in tables and graphics output from the conducted experiments to validate the proposed model. At this point, we judge it helpful to introduce concepts of statistical analysis. The central concept we must explain is hypothesis testing [56]. Suppose we formulated a hypothesis H and want to validate it through an experiment. We designed and executed our experiment and

obtained a set of data S, and then we refine H claiming that if the mean of S, $\mu(S)$, is inside a confidence interval $[h1, h2]$, then H is true. Since we observed a variation in data, we are not convinced about the validation of H, and we want to know if our data does not mean something completely different.

What we can do is apply some statistical tests on our data under H, and one way of doing it is using the null hypothesis significance test, the p-Value [56]. A p-Value is a statistical test that allows us to know the probability of our data confirming the negation of H, which we denote as the null hypothesis $\neg H$. The p-Value represents the probability of the data S come to confirm $\neg H$, and usually, values as small as 0.05 are considered safe enough. Sometimes we are interested in comparing the average return obtained from two distinct methods. For doing so when the sample size is small, we apply a statistical test referred to Student's t-Test [57]. The Student's t-Test is a statistical treatment that allows us to verify if the difference between two means is significant under the Student's distribution. The t-Value is a measure of how big is the difference between the two metrics relative to the variation of the data. As big is the t-Value as unlikely is the null hypothesis. A t-Value is usually used along with the p-Value on hypothesis testing.

Assume that we want to compare the performance of two systems when subjected to the same workload. Following the procedure explained in [56], it is possible to apply the Student's t-Test mean difference comparison to compare the two systems. When comparing the systems A and B, let's define the H hypothesis as A performs better than B. Then we calculate the p-value and t-value in H. If the p-Value is small enough, p-Value lesser than 0.05, we look at the magnitude and sign of the t-Value. If the t-Value is positive, A performed better than B. If the t-Value is negative, B is better. The greater the magnitude of the t-Value, the more relevant is the difference between the compared systems. However, if the p-value is higher than 0.05, we cannot say that there is a difference between the performance of the two systems.

A necessary condition to Student's t-Test is that data follows the normal distribution. We can assure that this assumption applies to our data performing a goodness of fit check, i.e., a statistical treatment that will measure how far the expected distribution of our data is from the normal distribution. Two normality tests that apply to continuous domain data are the Kolmogorov-Smirnov (K-S) and the Shapiro-Wilk (S-W) [58]. Both K-S and S-W tests will calculate the probability of incurring an error rejecting the null hypothesis, i.e., if we accept that our data do not follow the normal distribution. Regarding that [58] says that S-W is more powerful in rejecting data normality, the size of the sample influences both tests' precision. However, second [56], the K-S test is specifically designed for small and continuous distributions.

Chapter 3

Related work

In this chapter, we present the selected related work concerning the problem of enabling live video transcoding applying heterogeneous and distributed devices. These works were selected over 2019 to 2021, searching using the queries "distributed transcoding", "real-time transcoding", "live video transcoding", and "crowdsourced transcoding" in Google Scholar[1] and CAPES Scientific Journal Gateway[2] search engines. Only papers that presented a fully defined architecture that was distributed and could employ heterogeneous devices were selected. As long as we know, this is the only work to propose applying a multi-agent architecture to face the distributed transcoding in open and heterogeneous environment problems. Both Google Scholar and CAPES can search in many scientific databases. Papers selected came from IEEE [3, 4, 5, 9], ScienceDirect [6], and ACM [8].

We organized the approaches following the classification proposed in [8]. The authors analyzed many distributed transcoding approaches and classified them into three classes depending on the location of the computational resources employed in transcoding. The three classes are cloud-based, end-assisted, and edge-assisted. In cloud-based solutions, transcoding is made by cloud servers adequate for intensive computation. In end-assisted approaches, end-users devices are employed to alleviate cloud servers from part of the transcoding loading. Finally, in edge-assisted, some transcoding tasks are offloaded to edge nodes. All selected papers in this chapter classify whether as end-assisted or edge-assisted approaches.

Since each work has a different approach employing a specific method, and authors validated their approach in distinct scenarios, it is hard to compare them in terms of performance. Instead, we analyze the proposed solution searching for opportunities where T&RM could be applied to bring advantages to providers and end-users.

[1] See `https://scholar.google.com`.

[2] See `https://www.periodicos.capes.gov.br`.

3.1 End-assisted approaches

The work of Chang et al. (2016) [3] proposes a real-time distributed transcoding system for reducing latency in a closed surveillance system that can employ heterogeneous computers in a private network. Authors propose to use the Apache Storm[3], an open-source distributed system for data stream processing, to coordinate the transcoding work in real-time. The devices responsible for transcoding are statically assigned, although authors present a dynamic node assignment method to transcoding jobs as future work. Figure 3.1 illustrates the proposed work topology.

Since transcoder machines belong to the same company, the authors of [3] do not deal with open environments or untrustworthy agents. Despite this, a medium-large corporate network should have many idle devices proper to receive transcoding jobs. The proposed solution could benefit from applying T&RM to evaluate and select candidates to receive transcoding jobs, forming a pool of readily available resources. Having this pool in hand should make the system more scalable and reliable.

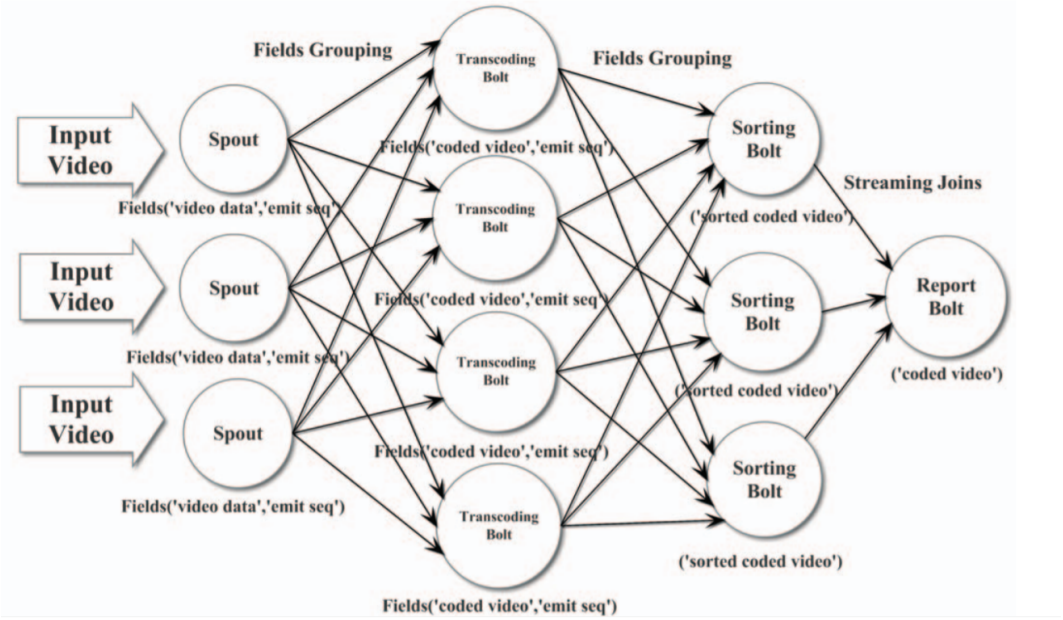


Figure 3.1: Storm-based video transcoding topology [3].

In He et al. (2017) [4] is investigated a FEC distributed transcoding scheme. The goal of the work is to minimize the delay experienced by the viewers. The system selects candidates to be transcoding nodes among the viewers of a stream section. The selecting criteria are stability online, i.e., how likely they are to stay online until the end of the streaming event. Then, the system organizes the selected nodes in a pool. The pool is a

[3]See `https://storm.apache.org/` for more information about Apache Storm.

B+tree data structure, which guarantees that inclusion and exclusion occur in $O(logN)$ and selecting the most preferred node for transcoding can take only $O(1)$. Some other heuristics for node selection are suggested, like video quality, but not further explored in the paper. In terms of architecture, regional data centers are responsible for distributing the transcoding task to candidate viewers. Figure 3.2 shows how the proposal defines the architectural elements in the FEC. Middle-edge regional servers manage the pool and the transcoding jobs assignments, while viewer's outer-edge devices perform the transcoding of the stream to lower bitrate representations. Although intended for the open FEC environment, the authors do not mention how to deal with untrustworthy transcoders. Besides, stability metrics should not suffice to characterize a viewer's device as a good transcoder. The device can be executing other tasks while simultaneously transcoding, which would affect processing time and bandwidth availability.

A complete evaluation scheme for [4] could combine stability online with other QoS-related metrics, such as transcoding time, network speed, and latency. If the system served those metrics to a T&RM, the produced trust values would represent how confident the system is that the selected node will contribute to overall system performance.

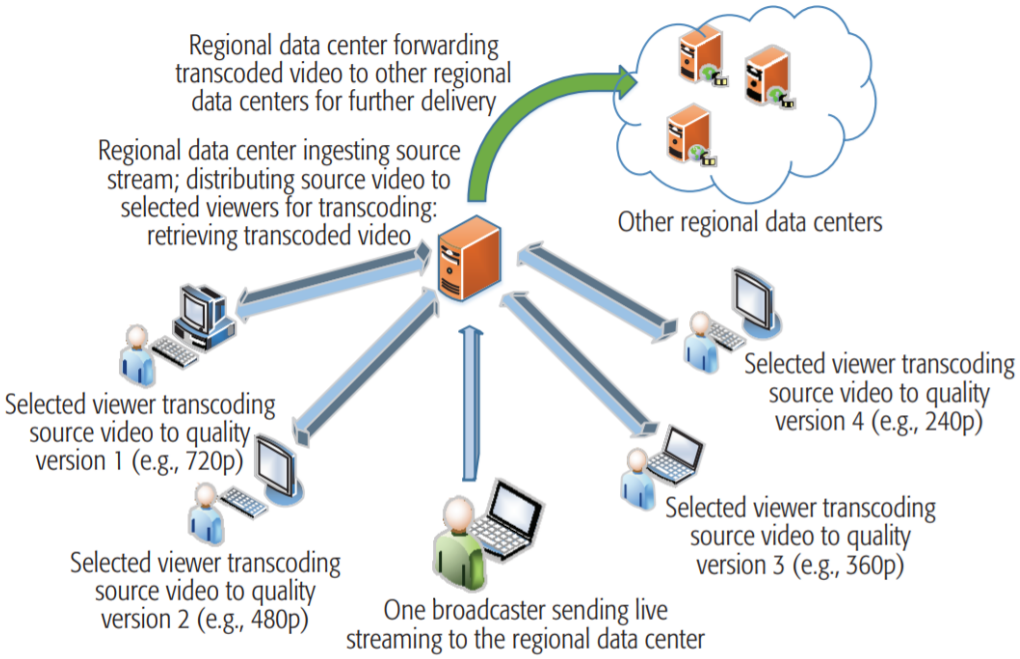


Figure 3.2: Fog based transcoding framework [4].

The objective of Liu et al. (2019) [5] is maximizing total network utility, pondering end-users QoE and network costs. The model utility function considers the viewer's expected QoE while watching the stream from transcoder nodes, which the proposed system selects among viewers and edge nodes previously available in a pool. The cost part is the success rate of devices transcoding the stream, which should reflect the node

stability online, as in [4]. Thus, authors take constraints and model them as a non-convex integer programming problem, which they solve by applying Complementary Geometric Programming (CGC). After that, the authors relax the CGC solution to a more feasible sub-optimal algorithm. Figure 3.3 shows an overview of the proposed system, showing the pool of transcoders reinjecting the transcoded stream into the CDN layer.

Once more, despite proposing to employ nodes in the FEC, the authors do not offer a strategy for dealing with untrustworthy agents. The assumption that all agents are trustworthy in an open environment is weak since the proposed incentive reward to transcoders can lead agents to selfish behavior. Therefore, the proposed system in [5] would benefit from adopting a trust and reputation approach to look for protection against possible harmful behaviors, where reputation evaluation must have a determinant role.

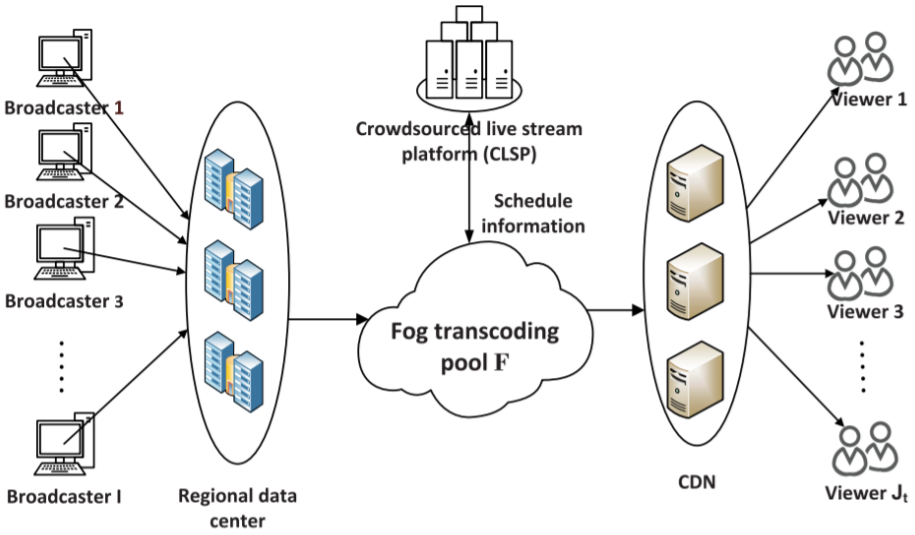


Figure 3.3: System model showing stream flow in the Fog [5].

3.2 Edge-assisted approaches

The objective of Bilal et al. (2019) [6] work is to minimize the backhaul utilization and the number of CDN hits per stream. The authors intend to achieve this by offloading the caching and transcoding jobs to Mobile Edge Computing (MEC) in Antenna Integrated Radios (AIR). Since MEC-AIR nodes are in the middle edge, closer to end-users than CDN servers in the inner edge, video requests will hit them first. If MEC-AIR nodes do not possess the requested bitrate version, they can obtain it from near nodes or, if necessary, request the higher bitrate version available from CDN and transcode it themselves. The most suitable bitrate is chosen by applying a greedy algorithm, which resulted from the

relaxation of an integer linear programming solution. Figure 3.4 shows how tasks are coordinated in the proposed network architecture.

We infer that the authors of [6] chose to employ MEC-AIR nodes since they are trustworthy by rule. End-user devices, for their turns, are heterogeneous and not trustworthy at prior. However, the same approach could be applied to minimize MEC-AIR hits if end-user devices were employed. Once more, the application of T&RM to compute trust and reputation values could devise between adequate and not-adequate end-users devices.

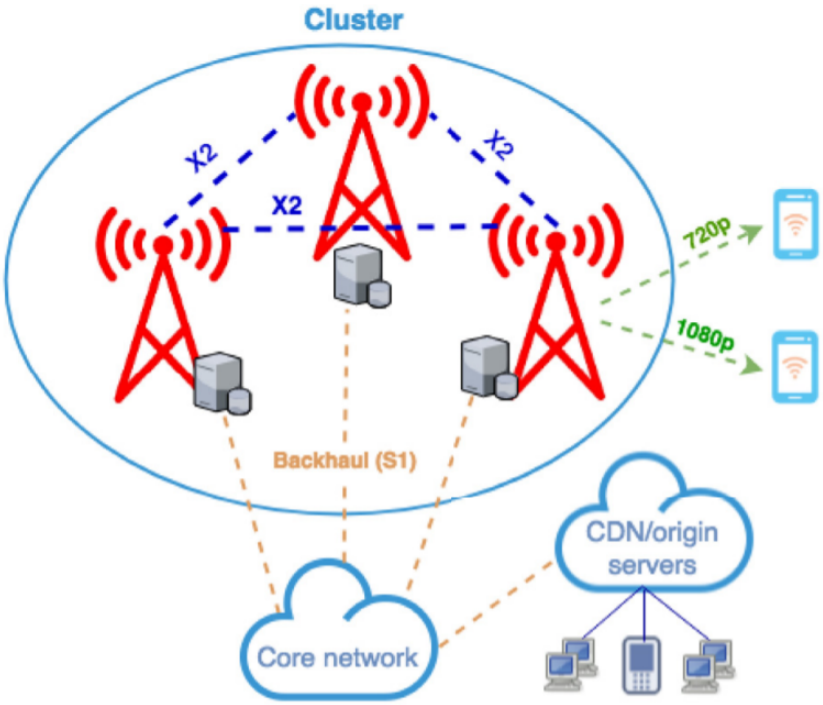


Figure 3.4: Caching and transcoding in middle-edge [6].

The objective of Fu et al. (2020) [7] is to cope with the problem of distributed transcoding in vehicular fog computing. As shown in Figure 3.5, the proposed solution employs stationary servers near Road-side Units (RSU) and mobile nodes installed in buses. The authors propose jointly optimizing end-user QoE and network parameters, deciding, for every vehicle, what bitrate representation should be provided and from what node to obtain it. The authors applied Deep-reinforcement Learning (DRL) and Actor-critic to solve the problem modeled as a Markov Decision Problem.

In [7], the end-users are the vehicles, but the authors considered them only as consumers, not producers, unless for the buses with an edge node installed on them. Considering devices in vehicles, passengers' smartphones and notebooks, would be helpful. A passenger on a long bus journey in a crowded roadway could collect some reward by sharing his processing power and network interfaces while entertaining himself at the same

time. Trust and Reputation values would indicate which of those devices to use and which not.

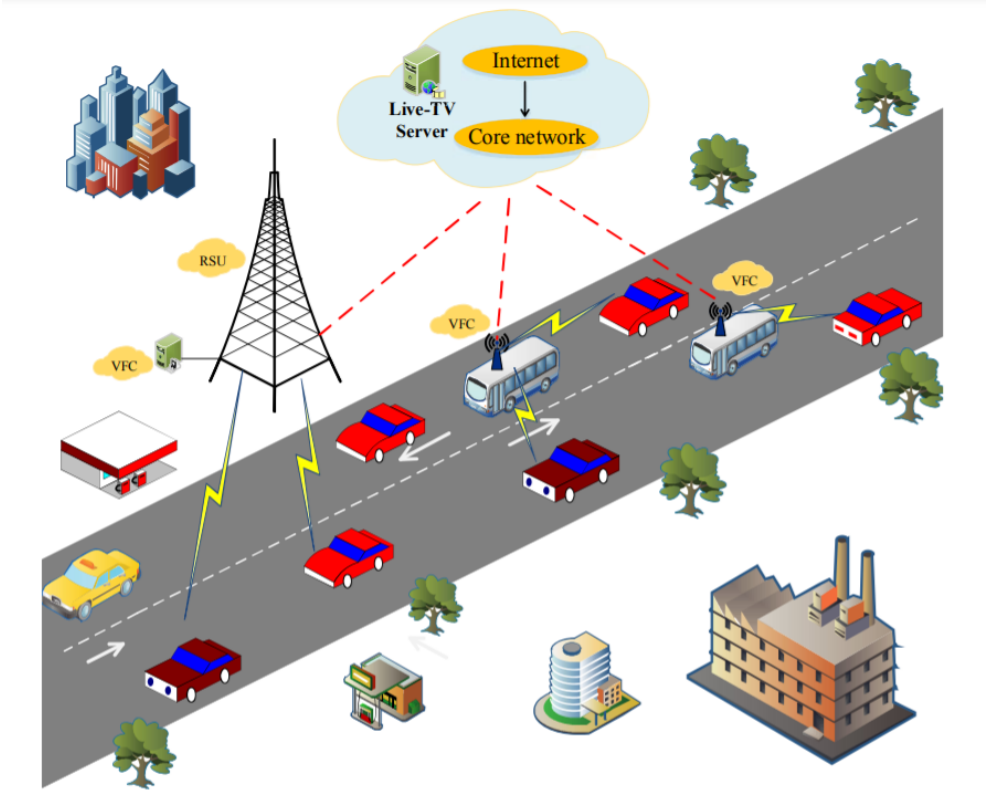


Figure 3.5: Transcoding in vehicular Fog Computing [7].

The proposal in Wang et al. (2021) [8] is a framework named ELCast to delegate the transcoding of lower bitrates to edge nodes and associate edge nodes with end-users. Moreover, the authors propose massively profiling end-users using DRL and Actor-critic to accommodate personalized QoE parameters in end-user to edge node assignments. The profiling should be executed at edge nodes since they state that the new edge architecture provides better hardware foundations for deep learning running at the edge. Figure 3.6 presents how the learning model interacts with the FEC to achieve the objectives of the proposal.

The authors of [8] also analyze other distributed transcoding approaches and classify them into three classes, depending on where are located the computational resources employed in transcoding: cloud-based, end-assisted, and edge-assisted. They advocate that edge-assisted approaches are the most advantageous. They argue that the uncertainty associated with end-user devices should turn transcoding services unstable. However, T&RM are adequate to mitigate the risk associated with that uncertainty, guiding the association to the most trustworthy devices.

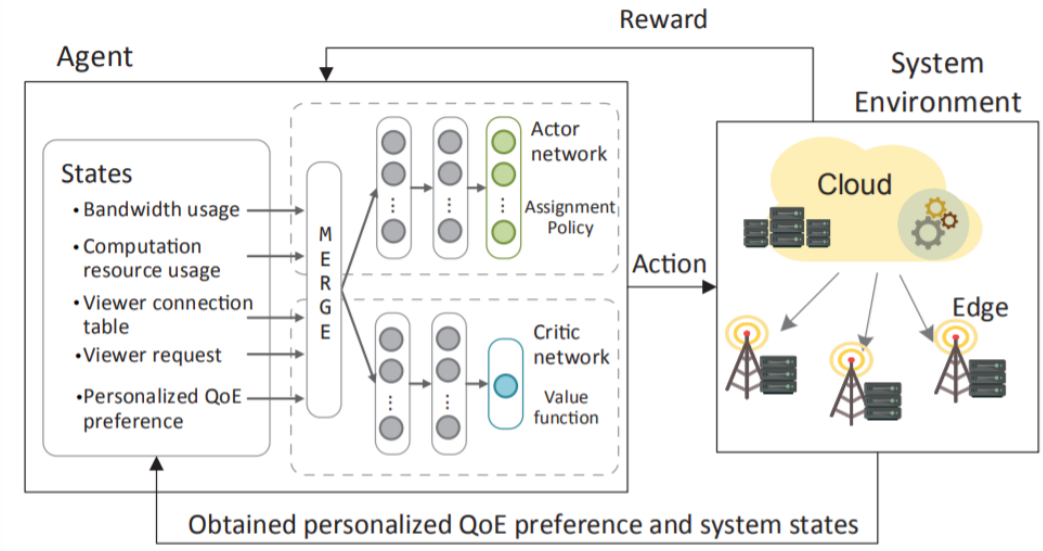


Figure 3.6: DRL, Actor-critic, cloud and edge nodes interaction [8].

Authors of Chen et al. (2021) [9] target optimize transmission resources (bandwidth) and transcoding resources (CPU) to improve QoE measures. Their approach employs cloud, edge, and end-user devices in transcoding tasks. They modeled the problem as two augmented Queues, one for transmission and the other for pending transcoding tasks, then an optimal solution is constructed using Accelerated Gradient Optimization. As shown in Figure 3.7, from provider to the viewer, the request bitrate representation passes through a route over the overlay network, which would involve nodes in every FEC layer.

Although employing end-user devices, nothing is said in [9] about how to deal with untrustworthy or erratic devices. This way, we inferred that the stochastic capacity of transcoders, like CPU processing power and bandwidth, must be somehow previously known. The solution should be complete with the application of an exploratory algorithm to learn about end-user device capacity available in a pool. Jointly to that, the application of T&RM trust values should cope with the risk of association with not well-known partners.

3.3 Final Considerations

Although it is hard to compare the presented related work employing different approaches and specific validation methods, we present Table 3.1 (increasing publication date order) to summarize important aspects considering the architecture proposed in this work.

Notably, related work missed approaching an open environment with its challenges, e.g., the problem of collaborating with potentially untrustworthy agents is prominent when

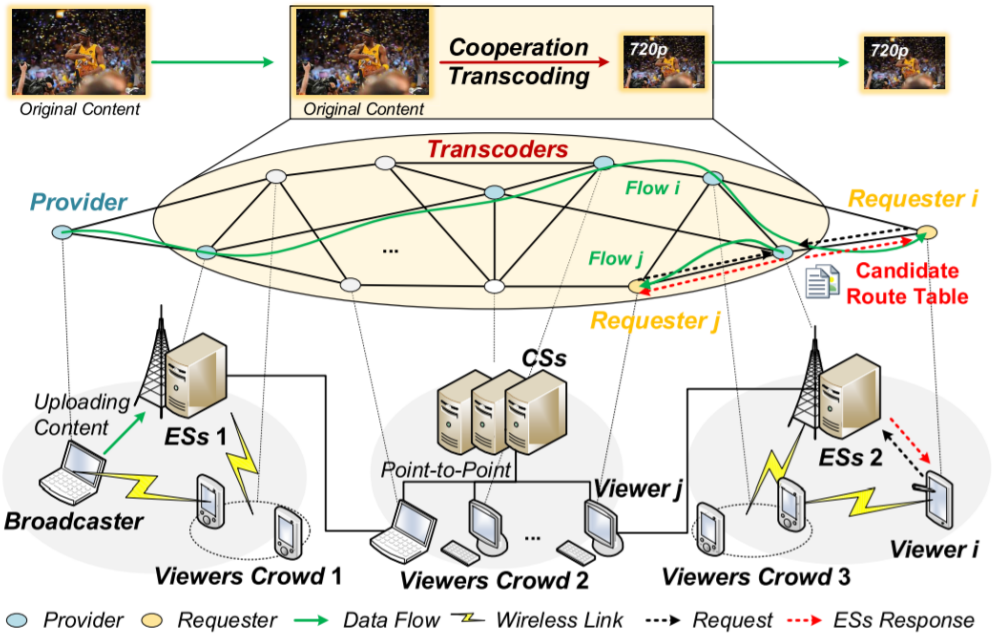


Figure 3.7: DRL, Actor-critic,cloud and edge nodes interaction [9].

end-user devices are employed. In our work, transcoding jobs will be dynamically assigned to nodes in every turn according to their past-observed performance. For doing so, we propose that T&R metrics represent an overall evaluation of utility contribution, not only including the node stability but also their competence for doing the assigned task. We also propose that transcoding should be done in the outer edge (nearby viewers' locations), allowing transcoded video segments to be served within relatively low latency. Besides, since we are applying T&R models that weighted witness credibility, our approach offers protection against fake feedback attacks from untrustworthy agents. Thus, we addressed the problem on three fronts: defining a multi-agent architecture, reasoning about selecting appropriate nodes for transcoding jobs in an open environment, and evaluating witnesses' credibility to avoid being influenced by untrustworthy agents.

In Chapter 4, we propose a multi-agent architecture that considers the questions presented in these previous works and also a scenario where many agents involved are autonomous. We believe that the architecture is adequate for the applications presented in this chapter. Additionally, the multi-agent architecture could improve these applications enabling them to use end-user devices safely.

Table 3.1: Related work comparison overview.

Reference	**Classification**	**Objective**	**Method**	**Dynamic assignment**	**Open environment**	**Employ end-user devices**	**Deal with untrustworthy partners**
Chang et al. (2016)	End-assisted	Reduce latency	Static assignment			✓	
He et al. (2017)	End-assisted	Minimize delay	B+tree sorting	✓	✓	✓	
Liu et al. (2019)	End-assisted	Maximize total network utility	Complementary geometric programming	✓	✓		
Bilal et al. (2019)	Edge-assisted	Minimize CDN hits	Integer linear programming	✓			
Fu et al. (2020)	Edge-assisted	Jointly optimize QoE and Network Parameters	Markov Decision Problem, DRL, and Actor-critic	✓	✓	✓	
Wang et al. (2021)	Edge-assisted	Accomodate personalized QoE parameters	DRL and Actor-critic	✓			
Chen et al. (2021)	Edge-assisted	Optimize bandwidth and CPU utilization	Accelerated gradient optimization	✓	✓	✓	
This work	End-assisted	Safely employ end-user devices	T&RM and MAB	✓	✓	✓	✓

Chapter 4

Multi-agent architecture

In this chapter, we detail the multi-agent architecture for coping with the distributed transcoding problem. The architecture comprises three well-defined agent roles:

- The Viewer Proxy is the agent role for the audience of the live video stream. This role is for those agents that ask brokers for the adapted ABR stream.
- The Transcoder role is for those agents interested in receiving transcoding jobs and being rewarded for them.
- The Broker's role responsibility is to manage the association of Viewer Proxies and Transcoders for the benefit of both.

Concerning the network architecture, it is necessary to point out that nodes must be geographically close to one another to keep latency inside an acceptable range. Thus, we suggest that viewers and transcoders reside on the same layer of FEC, the outer edge. Nodes performing the Broker role can be either end-user devices or FEC infrastructure components. A customized selecting algorithm seems necessary, and two are presented based on reputation, called ReNoS and ReNoS-II.

4.1 Requirements

The Tropos methodology described in Section 2.1.2 is used to elicit requirements of the multi-agent architecture and some aspects of agents' interactions. The sections in the sequence present the results of applying Tropos methodology to distributed transcoding in ABR.

4.1.1 Early requirements

As said in Section 2.1.2, the objective of the early requirements phase is to analyze the system as-is. We started identifying the roles and relations found in the streaming environment. Two actors were identified, the Viewer and the Streaming Platform, and the Streamer agent. Figure 4.1 shows these roles and their hard and soft goals. A hard goal of the Streamer agent is to make its content available to viewers. There must be a way for viewers to reach the content the Streamer produces. To achieve this, the Streamer relies on the reach of the Streaming Platform. Once the content can be located and consumed by viewers, the Streamer needs to engage its audience providing new content within an adequate frequency. According to [59], the uses and gratification theory can explain how engagement in media-consuming habits relates to various psychological needs.

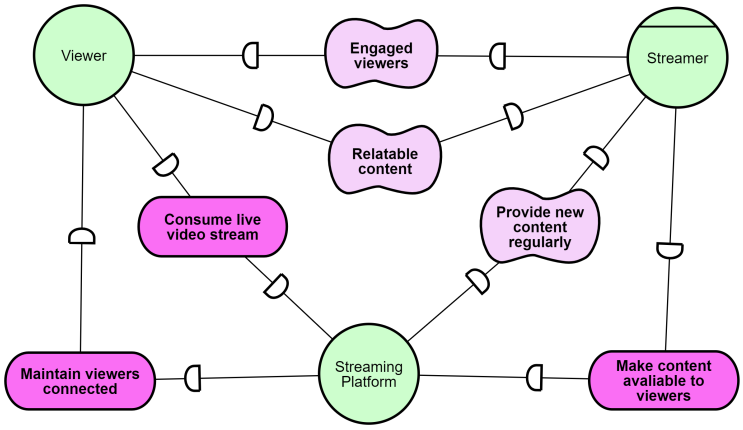


Figure 4.1: The streaming environment early requirements.

The Viewers' hard goal is to consume live video content from its preferred streamers, what they seek out from the Streaming Platform, among other ways of social recommendation. As explained before, viewers want to hit in content that they could relate with their own needs. Thus, they expect the streamer to produce content of its interest.

The Streaming Platform wants to maintain the viewers connected as long as possible. However, this is not possible without the Streamer's work. Therefore, the Streaming Platform must have the Streamer providing new content regularly. Since it is hard to state how frequently new content must be added to the platform to maintain viewers' interest, we listed this goal as a soft goal of the Streaming Platform, which depends on the streamer's productivity but not only. A bad QoS can make viewers abandon the presentation and disconnect, hurting the Streaming Platform objectives.

4.1.2 Late requirements

In the Late requirements phase of the Tropos methodology, the model defined in the Early requirements is updated with an actor to represent the system to be developed. The actor introduced is the Distributed Transcoding System (DTS). The DTS will act as an intermediary between Viewers and the Streaming Platform. Figure 4.2 shows the relations between the DTS and Viewers, and the DTS and the Streaming Platform.

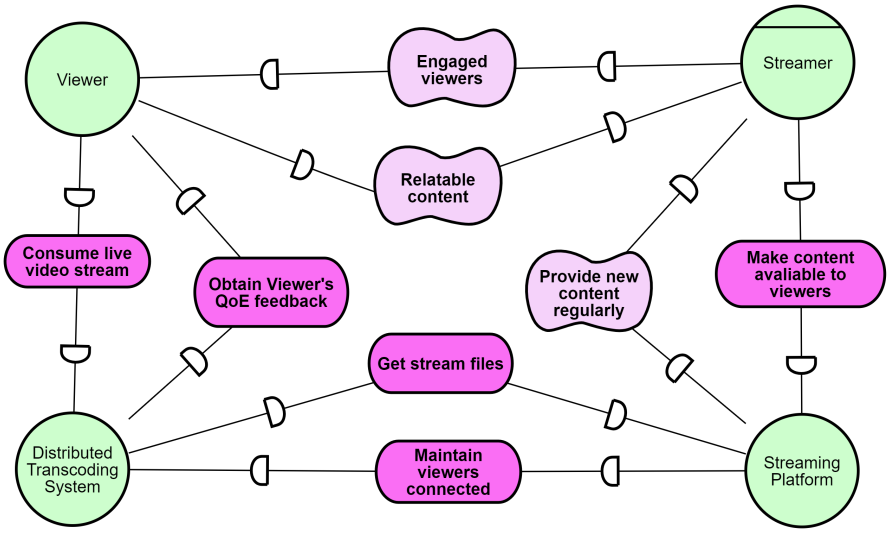


Figure 4.2: The streaming environment late requirements.

As an intermediary, the DTS actor will provide the viewers with the live video content they want to consume with an advantage: it will manage to ensure that viewers could get the better QoS possible, providing all the bitrate versions needed. In exchange, the DTS expects that viewers give feedback on the quality obtained from each video segment.

On the other side of the relations, the Streaming Platform depends on the DTS to maintain the viewers connected. The Streaming Platform relies on the DTS' ability to coordinate transcoding jobs so that the viewers obtain a good QoS. The Streaming Platform can evaluate the DTS performance just by observing the viewers' tendency of staying connected or not.

4.1.3 Architectural design

The architectural design phase is when the system introduced in the Late Requirements phase is divided into actors representing the software agent subsystems. In Figure 4.3, the DTS appears now divided into three roles: Viewer Proxy, Transcoder, and Broker. A role differs from an agent since it is a set of features observable from the exterior, and

an agent is an embodied entity. Since the same node can perform more than one role at once, these design elements do not classify as agents or actors.

Since the Viewer, a human agent, would not respond within the required time, we introduced the Viewer Proxy, a software agent represented by a role. The Viewer Proxy's responsibility is to interact with the other software agents using an interaction protocol for the sake of the Viewer. Those agents who will perform the Transcoder role are interested in receiving the transcoding jobs. As Viewer Proxy's role, the Transcoder role is also suited to software agents. The Brokers are up to select the better nodes to perform the transcoding tasks, negotiate with them the transcoding jobs, and then inform Viewer Proxies where to find the transcoded video segments. Within a specific geographic region, a Broker can coordinate at the same time the distributed transcoding of more than one live video streaming.

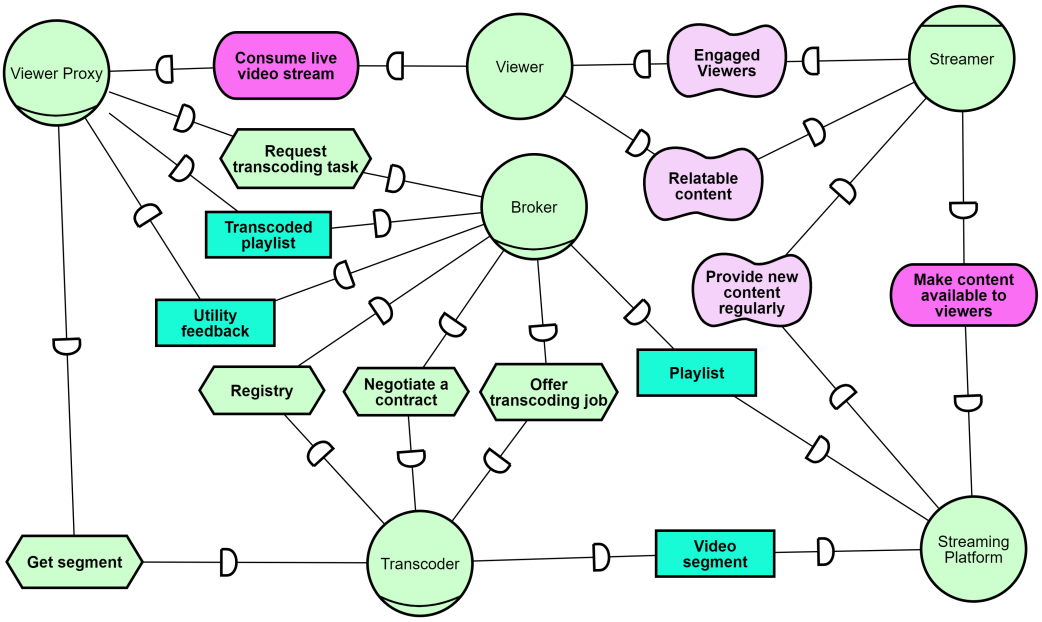


Figure 4.3: The DTS architectural design.

4.1.4 Capabilities

Capabilities are abstractions for agent's internal processes that allow them to fulfill a goal. In this section, we proceed to elicit the capabilities of each one of the roles defined in the architectural phase needs to have. Table 4.1 list the capabilities for the three roles in our architecture. In Table 4.1, we identify capabilities by a code compounded by two letters identifying the role (VC for viewer's proxy capability, TC for transcoder capability, and BC for broker capability) and a number. In the sequence, we describe them.

The Viewer Proxy capabilities are the following:

- VC1 - Require a broker to transcode the stream - when the Viewer Proxy identifies that the obtained QoS is not adequate, it can execute a plan to find a Broker to provide a transcoded version with more suitable bitrate versions.
- VC2 - Run playlist - this is a general plan with many actions related to the playback of the streaming video. It includes performing HTTP requests to obtain the playlist, the video segment and play them. After playing a video segment, the Viewer Proxy uses an assessment function[1] to evaluate the QoS.

Table 4.1: List of agents' capabilities.

#	**Role**	**Name**
VC1	Viewer Proxy	Require a broker to transcode the stream
VC2	Viewer Proxy	Run playlist
TC1	Transcoder	Register into brokers
TC2	Transcoder	Negotiate a contract with a broker
TC3	Transcoder	Accept transcoding job
TC4	Transcoder	Transcode video segment
TC5	Transcoder	Provide transcoded segment
BC1	Broker	Accept transcoding task
BC2	Broker	Obtain stream playlist
BC3	Broker	Provide transcoded playlist
BC4	Broker	Evaluate transcoders' trustworthiness
BC5	Broker	Accept transcoder registration
BC6	Broker	Negotiate contract
BC7	Broker	Assign a transcoding job

The Transcoder capabilities are the following:

- TC1 - Register into brokers - this capacity involves finding nearby brokers and selecting the ones into which the Transcoder will register.
- TC2 - Negotiate a contract with a broker - the Transcoder must evaluate the conditions offered by a broker in a contract, then decide if either they are acceptable or not.
- TC3 - Accept transcoding job - the Transcoder receives a request to transcode a video segment, then, after deciding whether accepting is convenient to its reputation, it accepts or rejects the job.
- TC4 - Transcode video segment - the Transcoder must require the original video segment from the Streaming Platform and then perform the required transcoding function.

[1] In Section 4.4, we present the Equation 4.5 to evaluate utility from the perspective of viewers.

- TC5 - Provide transcoded segment - when required by Viewer Proxy, the Transcoder has to provide the transcoded video segments it produced and stored.

Finally, the Broker role capabilities are the following:

- BC1 - Accept transcoding task - the Broker executes this capability when it receives a request for transcoding a stream from a Viewer Proxy.
- BC2 - Obtain stream playlist - to elaborate on the transcoding job offers and distribute them among the transcoders, the Broker should first obtain a playlist from the Streaming Platform containing the original video segment location. This segment information will be passed to the Transcoder when offering the transcoding jobs. The Brokers have to request playlists periodically because playlists are updated during the live event whenever new segments are available.
- BC3 - Provide transcoded playlist - when asked by the Viewer Proxy, the Broker must provide a playlist that allows locating the video segments distributed through the transcoders.
- BC4 - Evaluate transcoders' trustworthiness - periodically, the Broker must evaluate the transcoders' performance based on the feedback provided by Viewer Proxies. We suggest that QoS feedback will be given by Viewer Proxies piggybacking during the playlist requests to save communication costs.
- BC5 - Accept transcoder registration - this capacity will be activated when a Transcoder requires its registration. It involves two elementary actions, accepting the registration and updating the transcoders database.
- BC6 - Negotiate a contract with a transcoder - the Broker must select a registered transcoder without an active contract, evaluate that transcoder and then calculate the conditions upon which the contract will be offered.
- BC7 - Assign a transcoding job - this capacity involves selecting and executing a plan to choose appropriate transcoders within the registered and evaluated inside the pool of available transcoder, then managing the offer of transcoding jobs to them.

4.2 Architecture

In this section, we present the DTS architecture wherein the FEC layers including the nodes performing the three agent roles. The FEC architecture provides a macro vision

of the proposal. Then, we look inside the agents to suggest a micro layered architecture that enables agents to demonstrate their required capabilities.

4.2.1 FEC network architecture

The agent roles should rest in the FEC layered architecture described in Section 2.5, and presented in Figure 4.4. The diagram includes only a suggestion of network distribution since different roles can be performed by the same node. The roles were organized by the intended coverage area. The Streaming Platform lies in the core of the cloud. The Directory Facilitator is an agent whose responsibility is to serve the Viewer Proxies and Transcoders with the address of near Brokers. The Broker role agents can lie in the middle-edge or the outer-edge, so they can be reached by nodes in the outer-edge, not being too far from the clients.

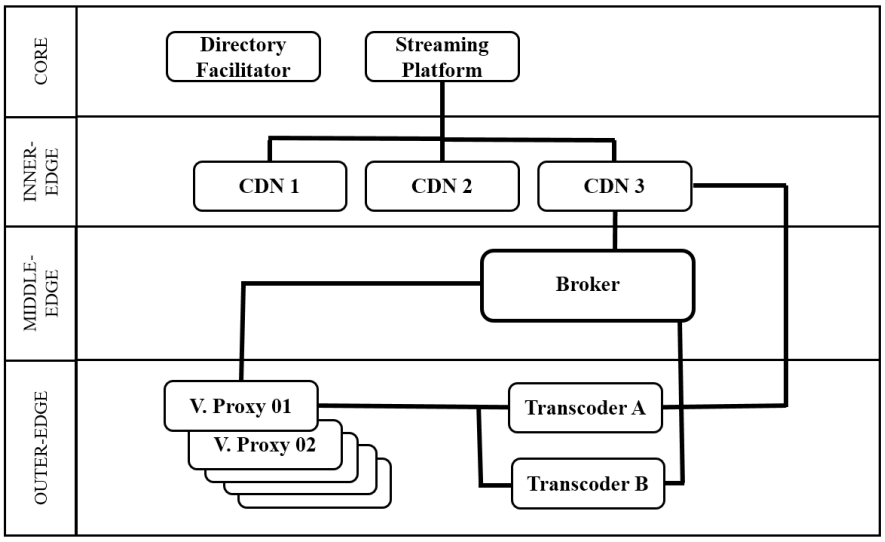


Figure 4.4: The agent-based architecture with the FEC layers.

4.2.2 Agent roles architecture

Since we propose to deal with video processing on the fly during live video events, we identify our approach as a real-time constrained system. During the transcoding of a live streaming session, our agents do not have much time for deliberation, which leads us to design agents where reasoning and performing are separated asynchronous processes. In the transcoding process, reactive behaviors are predominant. We suggest that two of our roles should be performed by hybrid agents. These are Broker and Transcoder roles. Viewer Proxy role can be performed by purely reactive agents.

The Broker role architecture is hybrid since one of the components is a T&R module. Figure 4.5 shows the internal architecture of the Broker role. The Messaging layer and the Belief Database Management are orthogonal to other layers allow them to operate asynchronously. The Belief Database is the repository of facts about the environment through which the layers exchange information about the world.

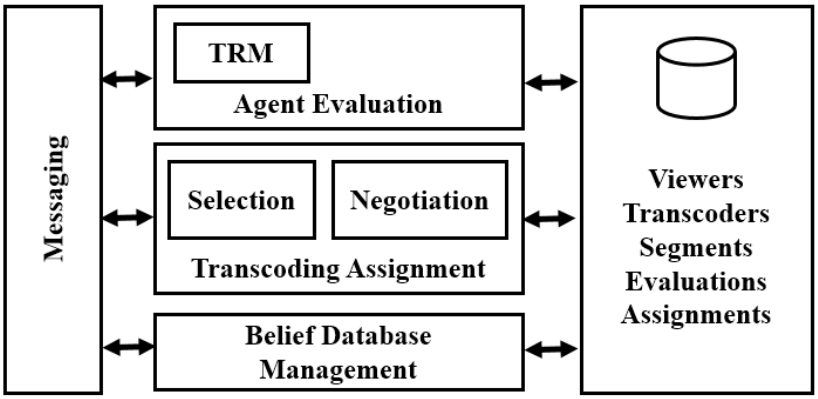


Figure 4.5: The detailed Broker's layered architecture.

The Belief Database Management layer is responsible for requesting the playlist from the Streaming Provider and registering the new segments, receiving the Transcoder registration requests, managing parameters as the reputation threshold and the transcoders' state, besides all other data repositories related tasks.

The Agent Evaluation layer will exchange the playlist with the reports of viewer proxies, converts them to ratings, and activate the T&R module, which converts those ratings in evaluations of trust values for Transcoders and witnesses credibility value for Viewer Proxies. Ratings and agent evaluations are stored in the Belief Database for future use.

The Transcoding Assignment layer has two responsibilities, negotiate transcoding contracts with transcoders and distribute the transcoding tasks to the available transcoders. This layer accomplishes these responsibilities by executing them asynchronously. The negotiation process's responsibility is to make agreements with all registered transcoders about the conditions for the transcoding tasks. Once the Broker and the Transcoder have a deal, the Broker includes the Transcoder into the available transcoders pool. The selecting process will periodically check for recently added segments. For every new video segment and desired bitrate, the Transcoding Assignment layer will use the Selection module to select a Transcoder for doing the job. Then, the algorithm will offer the task to the selected Transcoder. If a Transcoder rejects an offer, the segment returns to the pool of not-assigned segments. Table 4.2 shows how the Broker's layered architecture relates to the capabilities listed in Section 4.1.4.

Besides the orthogonal layers described in Broker's architecture, the Transcoder role layered architecture has two additional layers, as we can see in Figure 4.6. The responsibility of the Negotiation layer is registering the agent into Brokers and negotiating the transcoding contracts with them. The architecture is agnostic about how this layer can reason to maximize the return that a transcoder could obtain, although some approaches may require additional message exchange. For example, we do not preview Brokers disclosing reputation information with the Transcoder, but this could be useful. The Processing layer is responsible for requesting the original segments from the stream provider and transcoding them using the method negotiated with the Broker. It is also responsible for serving the transcoded segment files to the proxy viewers.

Table 4.2: Broker role – mapping layers and capabilities.

Layers	Capabilities						
	BC1	BC2	BC3	BC4	BC5	BC6	BC7
Belief Database Management		✓			✓		
Transcoding Negotiation	✓					✓	✓
Transcoder Evaluation			✓	✓			

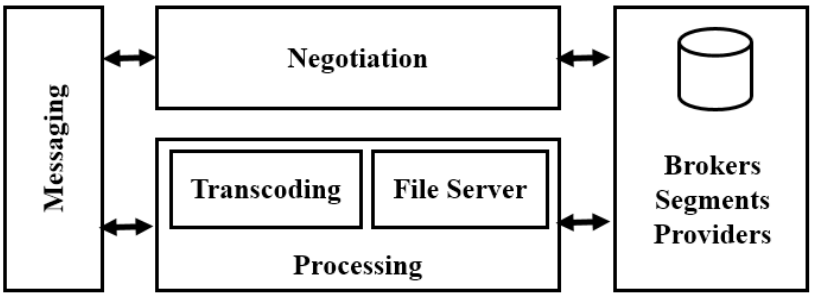


Figure 4.6: The detailed Transcoder's layered architecture.

Table 4.3 shows how the Transcoder's layered architecture relates to the capabilities listed in Section 4.1.4.

Table 4.3: Transcoder role – mapping layers and capabilities.

Layers	Capabilities				
	TC1	TC2	TC3	TC4	TC5
Negotiating	✓	✓	✓		
Processing				✓	✓

The Viewer Proxy role has a layer for reasoning about the QoE delivered to the viewer, as presented in Figure 4.7. The QoE Reasoning layer can proactively request

that a Broker arrange to make plus bitrate versions available. Hence, it needs to get information from the player and use it to invocate an assessment function to evaluate utility. In Viewer Proxy architecture, the player is also a layer that only communicates with the QoE Reasoning.

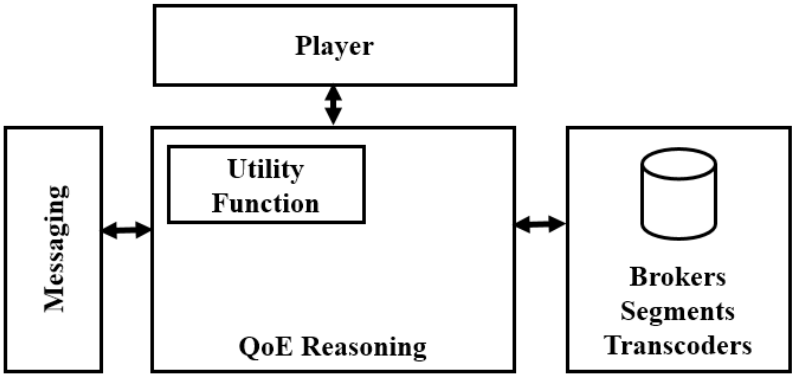


Figure 4.7: The detailed Viewer Proxy's layered architecture.

Table 4.4 shows how the Viewer Proxy's layered architecture relates to the capabilities listed in Section 4.1.4.

Table 4.4: Viewer Proxy role – mapping layers and capabilities.

Layers	Capabilities	
	VC1	VC2
Player		✓
QoE reasoning	✓	✓

4.3 Communication and interaction protocols

As software agents inhabit the FEC environment, the nodes performing the roles of our system must use the messaging layer to communicate with the environment and with other agents. Interfaces for receiving messages represent the agent's sensors, and those for sending messages are the actuators. Thus, receiving a message is a sense, and sending it is an action over the environment. In other words, agents performing their roles will perceive the environment interpreting the information received from other agents, then act by sending messages to drive the behavior of others. Table 4.5 presents a sample of the messages used as the communication protocol.

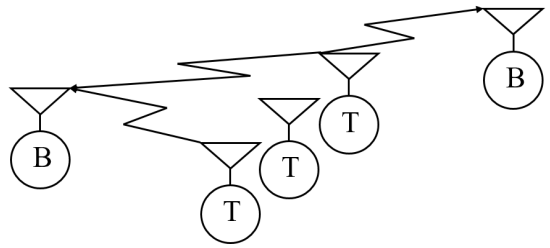
We are proposing that interaction among agents takes place in two distinct phases. The first phase is destined to brokers negotiate with transcoders the conditions under

Table 4.5: A sample of messages exchanged among agents.

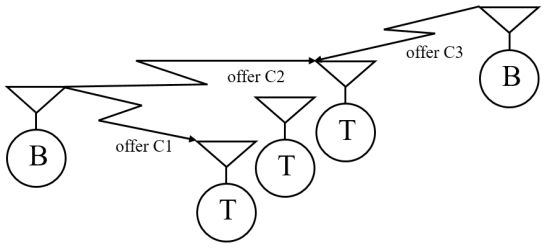
Message	Sender	Receiver
Propose transcoding a stream	Proxy Viewer	Broker
Accept transcoding a stream	Broker	Proxy Viewer
Reject transcoding a stream	Broker	Proxy Viewer
Request segment	Proxy Viewer	Transcoder
Provide transcoded segment	Transcoder	Proxy Viewer
Request transcoded playlist	Proxy Viewer	Broker
Request playlist	Broker	Streaming Platform
Propose a transcoding task	Broker	Transcoder
Accept the transcoding task	Transcoder	Broker
Reject the transcoding task	Transcoder	Broker
Request segment	Transcoder	Streaming Platform
Offer a contract	Broker	Transcoder
Accept the contract	Transcoder	Broker
Reject the contract	Transcoder	Broker
Inform that contract is finished	Broker	Transcoder

with the transcoding tasks will be distributed and rewarded. The second is when brokers, transcoders, and viewers' proxies will collaborate to enable which one to achieve their goals. We propose a protocol to govern interaction in each specific phase. However, before detailing the protocol, we present an overview in Figure 4.8.

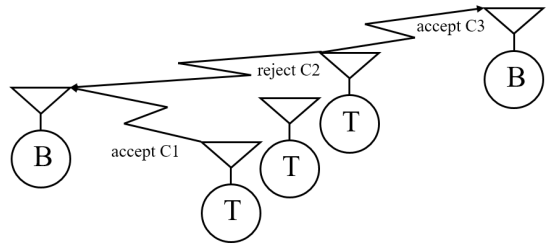
Figure 4.8 presents the core of agents' interaction in nine consecutive steps. We omitted message exchange between transcoders and streaming providers for better understanding. Transcoders register into brokers (Figure 4.8a), including that some transcoders can seek registration into more than one broker. After registering themselves and without active contracts, transcoders expect to receive offers. Thus, brokers offer personalized agreements to transcoders, depending on their reputation (Figure 4.8b). Transcoders respond if they accept or reject the offered conditions (Figure 4.8c). Already in the transcoding phase, brokers request the transcoding of video segments to selected transcoders (Figure 4.8d). So, transcoders can agree to or can refuse to do it (Figure 4.8e). A non-response is considered a refusal by brokers. Before knowing if transcoding is ready, brokers inform viewers' proxies where to get the transcoded segments (Figure 4.8f). Then, when they need them, viewers' proxies request video segments to transcoders (Figure 4.8g). In the course, viewers' proxies inform respective brokers of reward obtained interacting with transcoder (Figure 4.8h). Brokers take reports and use them to evaluate transcoders. When it is time, brokers inform transcoders that their contracts finish (Figure 4.8i).



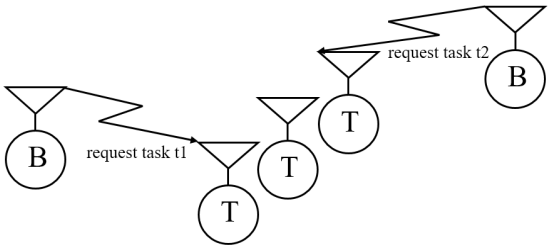
(a) Transcoders register into brokers.



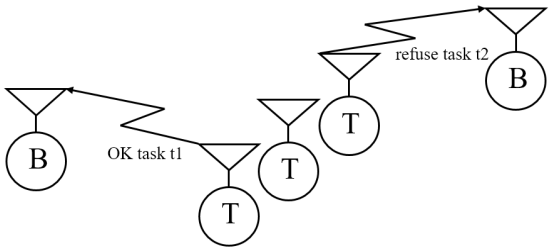
(b) Brokers offer contracts.



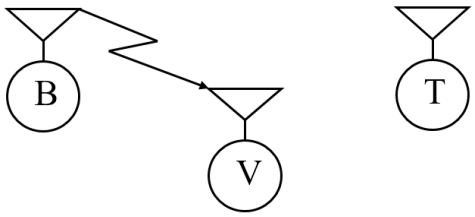
(c) Transcoders accept or reject conditions.



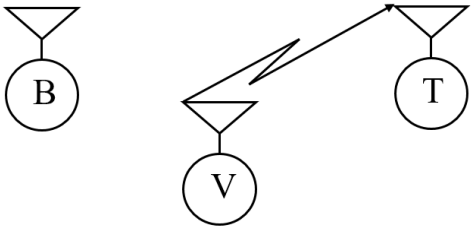
(d) Brokers request transcoding.



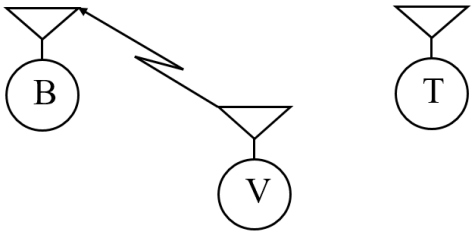
(e) Transcoders agree or refuse to do the task.



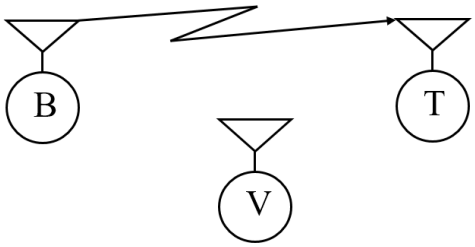
(f) Viewer's proxy receives segments locations.



(g) Viewer's proxy requests video segment.



(h) Viewer's proxy informs obtained reward.



(i) The broker informs current contract finished.

Figure 4.8: Roles interaction overview.

4.3.1 Transcoding phase

The sequence diagram represents interaction among objects in the Unified Modeling Language (UML), showing a temporal view of the message exchange [60]. Figure 4.9 is an example of interaction among the component roles of our proposed architecture and the Streaming Platform. The Viewer and the Streamer are human agents and interact in a wider time scale than the software agents in the diagram (omitted here).

Disregarding that some message exchanges can take place in parallel, the interaction in Figure 4.9 is an example of interaction among agents performing our architecture roles. The exchange of messages goes this way. The Viewer Proxy requests the transcoding of a live video stream to a Broker. Then, the Broker requests to the Streaming Platform the list of all video segments generated until that moment. After receiving the playlist, the Broker selects one of the registered transcoders to do the transcoding job. The Transcoder requests to the Streaming Platform the original video segment and transcodes it to the requested representation.

Meanwhile, the Viewer Proxy asks for the stream playlist to the Broker. The Broker responds with the location of the segments produced and stored by the Transcoders. The Viewer Proxy will download the transcoded segments from the Transcoder and then evaluate the QoS obtained using an assessment function. For managing to keep its internal buffer occupancy, the Viewer requests an updated version of the Broker's playlist with the new segments added while informing the Broker about the QoS of previously played video segments. The Broker will collect the Viewer Proxy feedback and then responds with the requested playlist.

4.3.2 Negotiation phase

An assumption we have made about FEC devices is that there are plenty of available devices, and those devices are interested in accepting transcoding jobs. Indeed, crowd-sourced approaches assume that viewers will borrow their computational resources for transcoding since they are interested in promoting the streaming channel. Nevertheless, we suppose that transcoders will perform transcoding tasks after some reward. Crowd-source supporters could be interested in receiving a VIP badge or some privilege that distinguishes them from the crowd. For other transcoders, incentives can be as simples as a certain amount of money.

In the proposed architecture, the Broker previously negotiate the conditions with the transcoders due to the narrow timeframe. Transcoders and brokers must agree about a contract that dictates conditions before the Broker includes devices in their available transcoder pool. The brokers guide their partnership with transcoders based on how

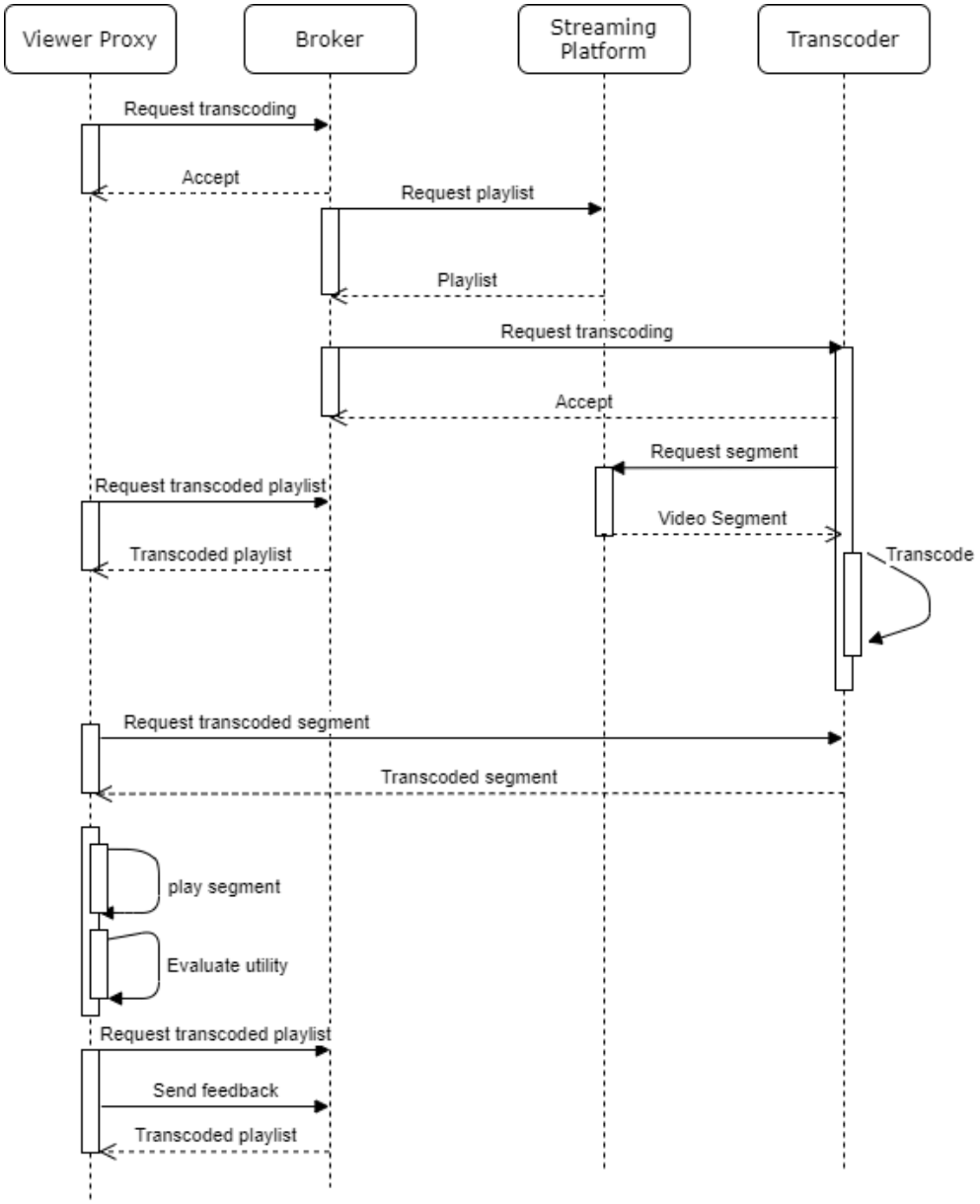


Figure 4.9: The UML sequence diagram representing agents' interaction.

trustworthy the Broker believes the Transcoder is. Likewise, we propose that the Broker reason over offered conditions upon these external believes.

In summary, the negotiation phase proceeds as follows. After a transcoder register into a broker, the Broker will reckon the pursuit information about transcoders. If the Broker and the Transcoder haven't previous interaction, the Broker will offer a default contract with a safe package of conditions to enable the Broker to evaluate the novice. If the Broker possesses previous beliefs of the Transcoder's trustworthiness, it will offer a customized contract. Thus, the more trust a broker has in a transcoder, the more it will offer tasks to that transcoder.

The contract conditions are the following:

- N: The number of transcoding tasks regarding the offer;
- P: The reward for every concluded task, referred to as the price;
- C: A compensation if the transcoder does not conclude any transcoding task, considering that it did not exceed the permitted number of refusals;
- Rs: The maximum number of times a transcoder can refuse a transcoding task without losing the compensation. If the transcoder fails to respond to an offer, the broker will account it as a refusal;
- T: The time the other conditions are valid is just after the broker receives an acceptance.

Now and then, the broker will inspect the database of registered transcoders looking after a transcoder without an active contract. The broker will avoid selecting those transcoders whose current trustworthiness is under a threshold. The broker will then calculate the customized conditions of the offer. After selecting a transcoder, the negotiation phase follows two simple steps, the broker offers the corresponding contract to the transcoder, and its responses either accept or reject the offer. Figure 4.10 presents the sequence diagram of the negotiation phase. Note that:

1. The Broker makes an offer to the Transcoder including the conditions $\{N, P, C, Rs, T\}$.
2. The Transcoder will answer whether it accepts or rejects the offer.
   (a) If the Transcoder accepts the offer, the Broker includes it into the pool of available transcoders.
   (b) If the Transcoder rejects the offer, the negotiation ends.
3. The broker informs the transcoder that the contract is finished when time runs out or the number of tasks is reached.

There is no prohibition that the broker selects a transcoder that has rejected a previous offer. There is no assumption about the transcoding reasoning of accepting or rejecting the Broker's offers. There is only one active contract between the same broker and transcoder, but no limitation about how much active agreements a transcoder could have since transcoders can register in more than a broker at once.

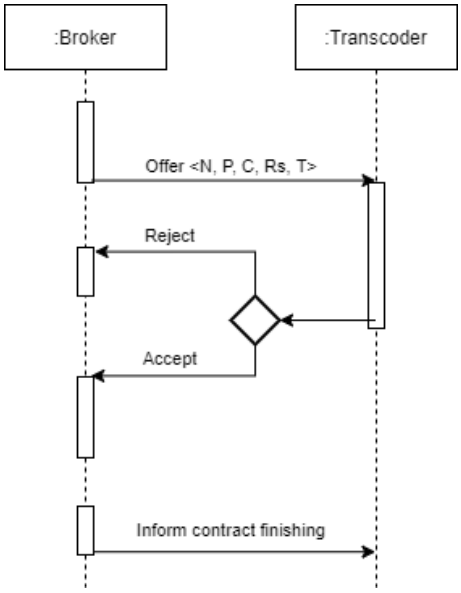


Figure 4.10: Sequence diagram of negotiation phase.

4.4 Model design

The multi-agent architecture design relates the performance measures for the three agent roles and the method to transform these measures into ratings adequate to be served for the T&RM employed. It also encompasses the selection algorithms ReNoS and ReNoS-II. Table 4.6 presents a summary of notations used in the model.

4.4.1 Contract conditions

The model encompasses the following procedures for customizing the contract's conditions to a specific transcoder. The conditions R and T should be empirically determined and be the same for all the offers. The condition P must depend on the effort to transcode original segments to transcoded versions, e.g., resizing from 1080p to 720p must afford a lower price than applying a CRF filter algorithm, which experiments shown is more computational demanding [24]. The Broker calculates the conditions N and C proportional to the transcoder trustworthiness using Equations 4.1 and 4.2, where τ is the trustworthiness of the transcoder tc, and $\{MaxN, MinN\}$ and $\{MaxC, MinC\}$ are value range for the conditions N and P, respectively. Unknown transcoders should receive the minimum possible values of N and C.

$$N(tc) = max(MaxN * \tau(tc), MinN) \tag{4.1}$$

$$C(tc) = max(MaxC * \tau(tc), MinC) \tag{4.2}$$

Table 4.6: Summary of notations.

Notation	Description
$PA(v, tc, s)$	Playback assessment, from Equation 4.3.
$A_V(v)$	Cumulative reward obtained by a viewer.
TC	Set of all transcoders.
S	Set of all video segments.
$A_T(tc)$	Cumulative reward obtained by a transcoder.
$R(tc, s)$	Reward that a transcoder receives per transcoded segment.
$C(tc)$	Compensation if transcoder is not selected to transcode.
$[MaxC, MinC]$	Valid range of compensation in a contract.
$A_B(b)$	Cumulative reward obtained by a broker.
$Rf(v)$	Refund a viewer pays to a broker by bps.
$M(s)$	Maximum reward possible of a segment, in bps.
$NR(tc, v, s)$	Normalized rating.
N	Number of transcoding tasks in a contract.
$[MaxN, MinN]$	Valid range of transcoding tasks in a contract.
P	Price paid by transcoded segment.
Rs	Max refusals which are tolerated by a broker.
T	Duration in which a contract is valid after acceptance.
τ	Value representing trustworthiness of a transcoder ($[0, 1]$).

4.4.2 Performance measures

In many aspects, the idea of the agents' rationality is linked to a performance measure [25]. Not only for evaluating the effectiveness of a plan of action but also for learning. When learning, the agents need a way to measure their performance. In the ABR domain, it is usual to resort to utility functions to evaluate performance, such as Equation 2.18 in Section 2.6. Thus, it is natural that the reasoning model of the three roles hangs on maximizing a gained utility.

Work on [14] defines an evaluation function that takes into consideration the average bitrate and the expected fraction of the time spent not rebuffering. The main goal is to optimize the utility for just one viewer, and it ignores the source of the video segments. We took those ideas, simplified the equation to a deterministic model, and adapted it to a distributed environment, allowing the viewers to evaluate the video segment providers with a performance measure related to their QoE.

However, before defining our performance function, let us take the video that is being broadcast and slice it into n segments with fixed duration T. We define a video stream S as a sequence of segments, $S = \{s_0, s_1, ..., s_n\}$. In the same way, we take the set of viewers in an ABR session as $V = \{v_0, v_1, ..., v_n\}$, and the set of transcoders as $TC = \{tc_0, tc_1, ..., tc_n\}$. $B(v, tc, s)$ is a primitive function that denotes the bits that a viewer v downloaded from a transcoder tc when it required a segment s, and $I(v, tc, s)$ denotes

the time interval passed during the playback of s, including eventual interruptions. The function $S(v, tc, s)$ says if the association was successful or not, i.e., if the viewer could completely download and play the segment, then it returns 1, or 0 otherwise.

Brokers have to counterbalance the benefits to viewers with the cost of offloading transcoding to the transcoders. In other words, viewers must pay transcoders accordingly with the reward they have got from them. Viewers might not pay for live video streaming, the same way the reward paid to transcoders is not actual money. Even so, it is helpful to pair viewers' and transcoders' rewards. This way, we assume there exists a function Rf that returns the value per bitrate a viewer must refund the broker so the broker can compensate transcoders for their job.

Using these characteristics, we define the Playback Assessment function (PA) as presented in Equation 4.3. PA returns the reward that a viewer obtained downloading a segment from a provider. Since all the sub-functions have the same arguments, we omitted them for readability. The constant β is empirically defined to adjust the equation considering how bad viewers evaluate video playback interruptions (QoE). The dimension of β is bps. The value returned by PA can be negative if the attempt to download and play a segment was not successful.

$$PA(v, tc, s) = Rf(v) \cdot \left[\frac{B \cdot S + \beta(T \cdot S - I)}{T} \right] \tag{4.3}$$

To understand how the PA equation should work, let us take the segment duration of 2s and β at $250bps$ and then compare a stream at 3,000 Kbps with another at 6,000 Kbps[2]. This way, switching versions will improve PA by 1%. However, if increasing the bitrate resulted in an interval of 1s per minute, the accumulated bitrate will decrease by 1%, then the viewer could have preferred the lower bitrate version. Figure 4.11 shows how the accumulated PA value is impacted by different β values when the time interval is increasing.

Viewer's proxy can use Equation 4.3 to evaluate rewards achieved cooperating with other agents' roles of our architecture. However, it is still necessary to link PA with the idea of the agent's utility. For doing so, let us take p as one of our proposed agent roles (broker, transcoder, and viewer proxy), and ar the rewards that an agent performing p accumulates until now. We assume that there is a utility function U that, given the situation where the agent performing p has achieved ar, returns the corresponding utility value. We also assume that U is monotonic between the ar domain, i.e., considering a fixed role p, if $ar_1 > ar_2$ then $U(p, ar_1) > U(p, ar_2)$. In other words, agents who perform roles defined in our proposed architecture will always prefer states that lead them to

[2]Using Twitch's encoder, the bitrates of 3,000 Kbps and 6,000 Kbps correspond to 720p-30fps and 1080p-60fps quality versions, respectively. See `https://stream.twitch.tv/encoding/`.

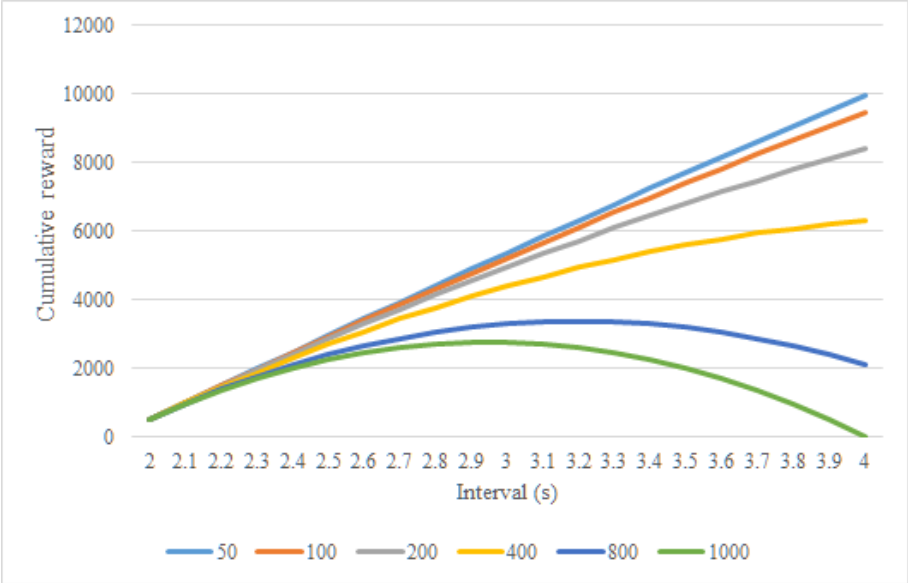


Figure 4.11: Accumulated PA value for different β values.

accumulate more rewards, and maximizing cumulative reward agents maximize utility as well.

Considering what was said, we proceed to define our roles' cumulative reward functions, which, under the assumption of the utility function U, we are using to compare the preferability of scenarios.

The Broker will reward the Transcoder based on the expected effort to complete the transcoding task. Equation 4.4 propose function R that takes as arguments tc and s, and returns the reward that tc achieved from transcoding s. If the Broker did not assign the transcoding of s to tc, R must return zero. Supposing a transcoder was not selected for any transcoding job, it receives compensation. The compensation is an incentive for transcoding reserving resources for completing future tasks. The function C returns a non-zero value if the Transcoder met the conditions.

$$A_T(tc) = \sum_{s \in S} R(tc, s) + C(tc) \tag{4.4}$$

Viewer's cumulative reward can be calculated from Equation 4.3 fixing a viewer and then iterating all over the transcoders and segments, then multiplying by the corresponding refund, as shown in Equation 4.5.

$$A_V(v) = \sum_{tc \in TC} \sum_{s \in S} PA(v, tc, s) \tag{4.5}$$

The broker's cumulative reward function A_B represented by Equation 4.6 is the difference between the sum of the rewards obtained by viewers (Equation 4.5) and the sum of the rewards delivered to the transcoders during a session (Equation 4.4).

$$A_B = \sum_{v \in V} A_V(v) - \sum_{tc \in TC} A_T(tc) \tag{4.6}$$

4.4.3 Ratings normalization

To evaluate transcoders' trustworthiness, brokers must collect feedback from viewers and use them as input to the T&RM. This procedure is prescribed by capability BC4 described in Section 4.1.4. The rating input for numerical T&RM are usually in the range $[-1, 1]$, so ratings obtained by Equation 4.5 must be normalized by the Broker. This normalization is a linear transformation from the range including the maximum and minimum possible reward to the interval $[-1, 1]$.

The viewer obtains the maximum reward when there is no interruption in the playback. In this case, the maximum possible reward is equal to the bitrate of the transcoded segment times the corresponding refund. Brokers can estimate the bitrate that resulted from the transcoding. Thus, we assume there exists the function M that takes a segment s and returns the estimated bitrate. However, the minimum possible reward depends on the timeout set up in the viewer's player, which is not on the broker's domain. In practice, players tolerate interruptions many times longer than the duration of a segment. Anyway, we assume that rating feedback less than five times the negative maximum has no use in the model. Putting these reasons on the table, we suggest that rating normalization use Equation 4.7, where TO is the timeout relative to the segment duration.

$$NR(v, tc, s) = max\left(2 \cdot \frac{PA(v, tc, s) - TO(s) \cdot M(s)}{(TO(s) + 1) \cdot M(s)} - 1, -1\right) \tag{4.7}$$

4.5 T&R for transcoder selection

Thinking about the work that our model imposes on brokers, after a certain point, brokers will form a set of available transcoders where some of them will be well known while others still need to be tested. Therefore, brokers should balance their strategies between exploiting the best-known transcoders or exploring the unknown ones to learn about them. Regarding learning of the available transcoders, we mean that brokers must form a belief about how good a specific node is as a transcoder to a group of viewers. This external belief, external since it is not reflexive, is a measure of the reputation of the transcoder in the community of viewers. For the Broker, a transcoder's reputation is represented by a pair of values - the transcoder's trustworthiness (T) and a measure of confidence in the first value, the reliability (ρ).

Brokers should be agnostic about the transcoders' expected performance to avoid making too many assumptions about transcoders' behavior. However, it can be helpful if brokers could model the probability distribution of transcoders' performance as a beta or a Gaussian distribution. Then, relate trustworthiness and reliability with the shape of the curve. Therefore, novice transcoders will begin with a short and wide curve. However, as the broker tests the novice, the broker accumulates information from the viewers, then reliability grows, and the curve narrows around the mean. Figure 4.12 is a representation of brokers model the transcoder performance as a single-tailed Gaussian distribution, where the first value of the pair is the trustworthiness and the second the confidence.

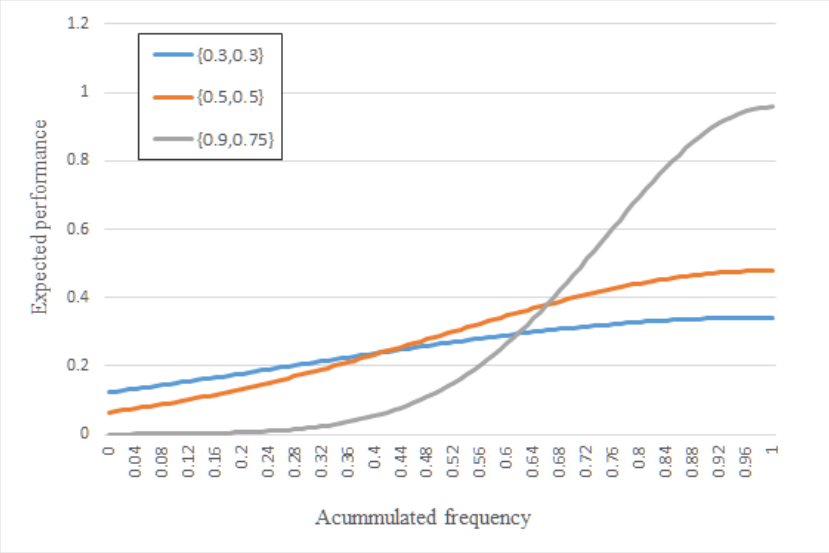
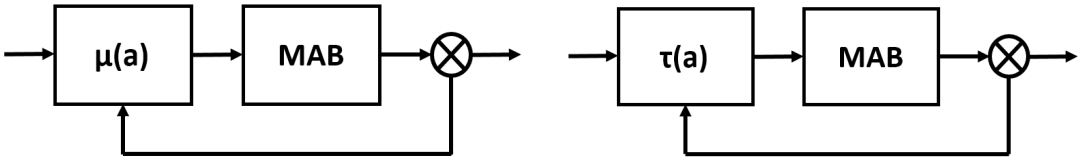


Figure 4.12: Proposed model for transcoders performance based on reputation.

Considering this transcoder reputation model, we suggest that the implementer of our architecture select a bandit algorithm to balance exploitation and exploration. The model is very close to the Thompson Sampling algorithm [61]. However, many stochastic bandit algorithms can fit well. Indeed, outcomes from experiments described in Chapter 5 indicate that combining reputation with stochastic bandit algorithms mitigates the risk of fake feedback attacks, also referred to as unfair feedback attacks. The conditions to this are that brokers must substitute simple averaging to reputation as input of MAB algorithms and apply a T&RM that offers witness credibility evaluation. Figure 4.13 is a representation of this idea. We hypothesize that relying on T&R capacity to evaluate witness credibility, stochastic bandit's algorithms should perform as expected even when attacked with fake feedback. Besides, since trust values in its bottom-line semantics is a subjective probability of performance, using $\tau(a)$ instead of $\mu(a)$ should not compromise the guarantees of stochastic bandit's algorithm.



(a) The conventional model. (b) The proposed solution.

Figure 4.13: Schematizing conventional MAB and T&R with MAB algorithms.

The FIRE T&RM describe in [34, 23], and in Section 2.2.3 of this work, fits in this proposal since it is a numerical model which offers a mechanism to witness credibility evaluation. As explained in [34], the FIRE T&RM only requires that Interaction Trust (IR) and Witness Reputation (WR) were available as information sources to enable the witness credibility, which is the way we propose to use it.

Nevertheless, for evaluating witness credibility, FIRE T&RM requires that report from direct interaction is available. Direct interaction means that the Broker has a player or emulates one, and it requires transcoded segments for itself. When the same node performs the roles of broker and viewer simultaneously, direct interaction is naturally available. It might not be possible when the Broker is an infrastructure node and not an end-user. In this case, the Broker must elect an accredited viewer per streaming session whose reports are as trustworthy as direct ones. Since not all T&RM requires direct interaction reports to evaluate witness credibility, we left the heuristic to electing an accredited viewer to the implementer of our architecture.

Reputation-based node selection algorithms

Since some MAB algorithms tend to narrow the set of choices to only a few options, it may overload some FEC nodes and degrade their performance. Besides, as FEC is a dynamic and unpredictable environment, even the best-evaluated node can quit without warning. Thus, MAB algorithms may perform better if modified to take into consideration load-balancing. In [24], we introduce the Reputation-base Node Selection (ReNoS), as presented in Algorithm 4.

The ReNoS algorithm balances exploration and exploitation to avoid overloading of the most trustworthy transcoders. When a new transcoder registers itself, it receives an initial trust evaluation above the threshold but slightly below the higher evaluation value, raising its chance to be selected in the next iteration. Concerning ReNoS parameters, the input *Nodes* is the set of available transcoders. The input *Factor* must be equal to or greater than 1 and represents how much is desired to explore the set of available

transcoders. The default value for *Factor* is 2. The input *Threshold* is the interaction threshold defined in [35] and explained before in Section 2.2 of Chapter 2.

Algorithm 4 Reputation-based Node Selection (ReNoS)

```
1:  procedure ReNoS(Nodes, Factor, Threshold)
2:      Update T&R evaluations of Nodes
3:      Sort Nodes by Trustworth
4:      Distribution ← Empty dictionary
5:      Probability ← 1.0
6:      for Each N in Nodes do
7:          if Trustworth(N) ≥ Threshold then
8:              Distribution[N] ← Probability / Factor
9:              Probability ← Probability − Distribution[N]
10:             Last ← N
11:     Distribution[Last] ← Distribution[Last] + Probability
12:     Selected ← Draw a node by Distribution
13:     return Selected
```

An issue with ReNoS is that it requires the initial trust value to be high. Thus, the trustor must believe that a new and unknown node is as trustworthy as a very known and well-evaluated one. Otherwise, the exploration will not work adequately. This policy is comparable with the UCB1 MAB algorithm, where the less tried bandits are those with the highest chance of being selected in the next round. Nevertheless, a high initial trust value might not be adequate for other usages than selecting. For example, when we introduced the negotiation phase protocol described in Section 4.3.2, the selection algorithm had to be changed so it would work with a more conservative initial trust value.

Thus, a new version of the ReNoS algorithm was defined as presented in Algorithm 5. The Reputation-base Node Selection II (ReNoS-II) is based on the same idea of the Thompson Sampling MAB algorithm, first proposed in [61]. The ReNoS-II takes into consideration the node performance modeling suggested in Section 4.5. When a new transcoder registers itself, it receives a reputation value with low trustworthiness and confidence values, so its performance distribution curve is flat and wide, close to the uniform distribution. ReNoS-II will use the set performance curves to predict the node's reward and select the node with the highest predicted value. Thus, with a flat and wide curve, a high reward is as probable as a low reward, then this algorithm will likely select the novices. The performance curve narrows around the confidence value as the confidence value increases, so veteran nodes are more prone to exploitation than exploration.

The inputs of ReNoS-II are the set of available nodes, *Nodes*; *ModelDistribution*, which is a probability distribution to predict reward based on trust and reliability values;

Trustworthiness, a function that returns the trust value of a node; *Reliability*, a function that returns the reliability of the trust value returned by *Trustworthiness*; and the *Threshold*. We have empirically determined that the best trust threshold below which brokers should disregard transcoders can be obtained by the equation $th(n) = 1 - 1/n$, where th is the desired threshold, and n is the number of available transcoders.

Algorithm 5 Reputation-based Node Selection II (ReNoS-II)

```
1:  procedure ReNoS-II(Nodes, Model-Distribution, Trust, Reliability, Threshold)
2:      Update T&R evaluations of Nodes
3:      selected-node ← None
4:      greater-prediction ← -1
5:      for Each n in Nodes do
6:          T ← Trust(n)
7:          ρ ← Reliability(n)
8:          if T ≥ Threshold then
9:              predicted ← Model-Distribution(T, ρ)
10:             if predicted ≥ greater-prediction then
11:                 selected-node ← n
12:                 greater-prediction ← predicted
13:     return selected-node
```

In Chapter 5, we validate the proposed architecture in real-time transcoding scenario. We believe that the architecture adequately copes with the problem of distributed transcoding on the FEC, as indicated by the experimental results.

Chapter 5

Experimental validation

Real-time transcoding of a live streaming event in ABR requires tight synchronization. Transcoders must download a part of the video stream before starting the transcoding, and only after they finish the job will the resulting segment be available for the viewers. The contribution of a transcoder settles in close relation to its processing power and network bandwidth. Notwithstanding, our approach must observe if transcoded segments are ready on time as requested.

In this chapter, we present the designed experiments to validate our proposal. The main objective of the experiments is to check whether the multi-agent architecture can adequately cope with the problem of distributed transcoding on the FEC, considering the risk of cooperating with potentially untrustworthy partners. More specifically, we want to verify the following assumptions:

1. Combining T&RM and MAB improves nodes selection by mitigating the risk of fake feedback manipulation.

2. The algorithm ReNoS-II can outperform the original ReNoS when initial trust values are far from the highest possible.

We intend to prove the feasibility of the proposed architecture using a prototype implemented strictly following the guidelines described in Chapter 4. The prototype performs simulations of the tasks needed in real-time distributing transcoding. In this experiment, we have to generate a stream whose video segments duration, periodicity, and length ensembles an actual live video streaming session. For doing so, we appropriate the experimental results presented in [24], which found out that viewer players request new video segments in a period close to the duration of the segments. Also, a typical end-user CPU should complete a segment transcoding in up to $0.5s$.

Validating our multi-agent architecture combining T&RM and MAB for selecting nodes in the FEC involves identifying the conditions in which the environment open-

ness could harm the MAB algorithms guarantees. One of these conditions is when agents are not trustful in their feedback reports. Since transcoders are interested in receiving as many transcoding jobs as possible, untrustworthy transcoders may form a coalition with viewers to manipulate brokers' choices. This kind of attack to trust systems is referred to as fake feedback attacks [41].

In [24], we compared the performance of the ReNoS algorithm against MAB UCB1 and random selection of transcoder nodes. Both ReNoS and MAB UCB1 performed better than random selection, presenting close results when the comparison involves only a cumulative reward. However, ReNoS has the drawback of needing the initial trust value to be close to the maximum for enabling the exploration of novices. Usually, initial trust and reputation values are set to the minimum or a neutral to discourage agents from switching identities. We designed the algorithm ReNoS-II so initial trust can be neutral. We validate the ReNoS-II algorithm by comparing it with Random selection, MAB UCB1, ReNoS, and the simple application of T&RM, which we referred to as Greedy trust (GT). We apply them in our simulated environment and compare the outcomes.

5.1 Simulation prototype

We developed a prototype to simulate the validation environment using the Java Agent Development Framework (JADE) [62]. JADE is an open-source framework for multi-agent systems development using Java programming language. JADE has the advantage of being compliant with the Foundation for Intelligent Physical Agents (FIPA)[1] specification and has built-in features that simplify the implementation of behaviors, messaging, and response to time events. We have chosen the JADE framework for implementing the prototype of the DTS due to our familiarity with the Java programming language and the JADE similarity with other relevant Java simulation tools, like PeerSim[2]. The prototype source code is available at [63].

Figure 5.1 presents the main classes implemented in the JADE prototype. All three roles are subclasses of Agent, one of the classes of JADE in-built class hierarchy. The Classes *Broker*, *ViewerProxy*, and *Transcoder* are abstract classes, i.e., they cannot be instantiated. They need to be extended by concrete classes that implement some specific behaviors. What can be seen about the abstract class *Broker* is that all brokers need to have a collection of segments and a collection of transcoders, and the *ViewerProxy* subclasses will have a player as a component.

[1]FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies. For more information, see `http://www.fipa.org/`.

[2]See `http://peersim.sourceforge.net/` for more information about PeerSim.

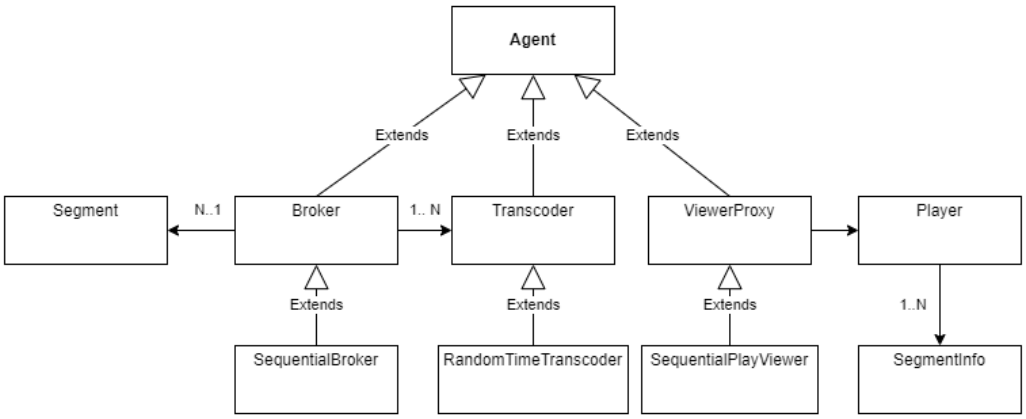


Figure 5.1: The simulation prototype class diagram.

Figure 5.1 shows the concrete classes *SequentialBroker*, *SequentialViewerProxy* and *RandomTimeTranscoder*. The prefix *Sequential* indicates that the segments are processed within their order of creation. The *RandomTime* prefix in the *RandomTimeTranscoder* means that the time spent transcoding a segment is randomly distributed inside an interval. The average interval is characteristic of the profiles listed in Table 5.1. Behaviors are not represented in Figure 5.1, but are instantiated in the concrete subclasses and are closed related to the capabilities specified in Section 4.1.4 from Chapter 4.

Table 5.1: Transcoder profiles.

Profile	Serving Range	Transcoding Range
A	$1s \pm 15\%$	$0.5s \pm 100\%$
B	$2s \pm 30\%$	$0.75s \pm 100\%$
C	$4s \pm 60\%$	$1s \pm 100\%$

5.2 Experimental setup

The experimental setup units and parameters are defined as follows. The broker agent can select nodes using five algorithms: Random selection, MAB UCB1, GT, ReNoS, and ReNoS-II. Using random selection, brokers select transcoders by change with equally distributed probability. When using the GT algorithm, brokers select the transcoder with the highest trust value after a fixed number of bootstrap turns. In some aspects, GT is similar to the ϵ-First algorithm described in Chapter 2, where the bootstrap turns correspond to the initial exploratory phase. The algorithms MAB UCB1, ReNoS, and ReNoS-II, will work as described in Chapters 2 and 4, respectively. Viewer Proxy agents will request segments at their duration, i.e., every 2 seconds. We set the Viewers' players'

buffers for three-segment length. Transcoders' performance is accordingly the profiles of Table 5.1. The serving range determines how long a viewer will wait for a segment, but transcoding times include time spent downloading original segments from the server. Transcoders of profiles A and B tend to accumulate positive results, but the C profile is more prone to harm viewers' utility. In Table 5.1, values are consistent with the range observed from the experimental outcomes in [24].

The simulated live video streaming generates new segments every 2 seconds. Every segment is 2s in length and has 1MB after transcoding, even though segments have different bitrates in real situations. The total video length is 400 seconds or 200 segments. In addition, agents are willing to exchange information and never reject an offer. Those are assumptions that we can hardly find in a real scenario. However, our objective is to compare the three algorithms in fair conditions. Besides, these assumptions are close to those used to test the FIRE model in [23]. Other experimental parameters are summarized in Table 5.2. For understanding the parameters *Alpha*, *Beta*, and Credibility threshold refers to Section 2.2 from Chapter 2.

Two scenarios were tested to make explicit the effect of fake feedback attacks and how they could compromise our distributed transcoding system if T&RM were not employed. As input to the algorithm MAB UCB1, we used the simple averaging of all normalized ratings (range -1 to 1). As input to GT, ReNoS, and ReNoS-II, we used the trust and reliability values produced by the FIRE T&RM [23] using the witness credibility approach proposed in [34]. For providing the direct interaction required for witness credibility evaluation, we introduced an additional viewer proxy agent referred to as the accredited viewer. The reward accumulated by the accredited viewer is not added to our comparisons.

The ReNoS-II algorithm receives as a parameter a PDF that characterizes the choices in terms of trustworthiness (T) and reliability (ρ). In these experiments, we used a Gamma distribution whose parameters $\{\mu, sd\}$ are defined by Equation 5.1.

$$\mu = T \cdot \rho, sd = e^{-2 \cdot \rho} \tag{5.1}$$

The two tested scenarios comprise:

- Scenario 01: The objective is to obtain a comparative baseline from fair evaluations. It employs twenty viewer proxies which are all honest about their feedback reports. Transcoders are referred to as $ta1$, $tb2$, $tb3$, $tc4$, and $tc5$ accordingly to its profile in Table 5.1, i.e., $ta1$ is of profile A, $tb2$ and $tb3$ are of profile B and $tc4$ and $tc5$ of profile C.
- Scenario 02: The objective is to observe what happens when a transcoder tries to manipulate the distributed transcoding with fake feedback attacks. In this scenario,

Table 5.2: General experimental parameters.

Parameter	Value
Number of repetitions	12
Confidence interval	95%
Number of Brokers	1
Number of Viewer Proxies	20
Number of Transcoders	5
Number of segments/turns	200
Segment duration	2s
Viewer Proxies' segment's buffer length	3
Viewer's player timeout	6s
Alpha	1
Beta	1,000
Credibility threshold	0.8
Evaluation values range	$[0, 1]$
Trust threshold	0.5
Initial trust value	0.75
Initial reliability value	0.5
ReNoS-II PDF function	see Equation 5.1
$Rf(v)$	$1,000$ per Mbps
$R(tc, s)$	fixed in 1 per segment
$C(tc)$	fixed in zero
GT bootstrap turns	40

we added two co-opted Viewer Proxy agents. In every opportunity, they evaluate the transcoder $tc5$ with the highest possible rating (1) and the other transcoders with the worst possible (-1). The goal of the untrustworthy agents is to increase the chance of $tc5$ being selected by the Broker.

5.3 Measurements

From data collected during the experiment, we derived three metrics used to analyze and compare the proposal performance. The metrics are Broker's cumulative reward, exploration factor, and transcoder $tc5$ assignments.

Broker's cumulative reward

This measure is the reward accumulated by the broker during the whole running, calculated accordingly Equation 4.6. In our architecture, the first step to obtaining this metric is a viewer's proxy applying Equation 4.3. Let us assume that the believes' database of a viewer proxy v_1 was the way described in Table 5.3. The playback interval is the

time spent by the viewer's player playing a segment, including in it the time when the playback was eventually interrupted. Success is 1 if the player could finish playing the segment, or 0 if the playback waited until it reached timeout.

Table 5.3: Example of a viewer's proxy believes' database.

Segment	Source	Length	Duration	Playback interval	Success
s1	ta1	1Mb	2000s	2000s	1
s2	tb2	1MB	2000s	2500s	1
s3	tc5	1MB	2000s	6000s	0

When it is time to request the updated playlist, v_1 reports to the broker the rewards it obtained from every played segment. Considering the parameters in Table 5.2, v_1 calculates playback assessment as follows:

$$PA(v_1, ta1, s1) = \frac{1,000}{1Mb} \cdot \frac{1Mb \cdot 1 + 1,000 \cdot (2,000 \cdot 1 - 2,000)}{2,000} = 0.500$$

$$PA(v_1, tb2.s2) = \frac{1,000}{1Mb} \cdot \frac{1Mb \cdot 1 + 1,000 \cdot (2,000 \cdot 1 - 2,500)}{2,000} = 0.262$$

$$PA(v_1, tc5.s3) = \frac{1,000}{1Mb} \cdot \frac{1Mb \cdot 0 + 1,000 \cdot (2,000 \cdot 0 - 6,000)}{2,000} = -2.861$$

We calculate the broker's cumulative reward as follows if v_1 were the only viewer registered into the broker, and segments listed in Table 5.3 are the only ones to consider:

$$A_B = (0.500 + 0.262 - 2.861) - (1 + 1 + 1) = -5,099$$

The cumulative reward calculated above is negative since the chosen parameters toughly penalize utility when playback is interrupted until timeout, which is the case with v_1 trying to play segment $s3$. However, an unsuccessful attempt to play a segment was a rare event in the experiment described in this work, which allowed us to obtain a positive cumulative reward in all repetitions.

Exploration factor

The exploration factor is the proportion of times the broker selected a transcoder other than the most selected. If the exploration factor is too low, it should indicate that the algorithm could overload a transcoder by over selecting it. Let us suppose that transcoder $tc5$ was the most selected transcoder after a repetition ends, and it was selected to transcode a segment 60 times. Since the total number of segments is 200, as described in Table 5.2, we can calculate the exploration factor as follows:

$$Exploration\_Factor = (200 - 60)/200 = 0.70 \quad (5.2)$$

Considering the number of available transcoders is 5, the maximum possible exploration factor is 0.80, and the minimum is zero. An exploration factor of 0.7 means that the most selected transcoder had 71.43% more task assignments than any other transcoder.

Transcoder $tc5$ assignments

This measure is the total number of times that transcoder $tc5$ was selected. An increase in this measure between Scenarios 01 and 02 should indicate that the fake feedback strategy was successful.

5.4 Outcomes

Table 5.4 presents the summarized data obtained from the experiment after all the repetitions. It shows the cumulative reward, the exploration factor, and the number of assignments the broker gave to transcoder $tc5$. We present the measures summarized by the average and standard deviation.

We intend to use Student's t-test to compare the averages in cumulative reward as it is advised by [56] for comparing two systems' performances. However, first, it is necessary to verify if we can accept that data follows the normal distribution. Figure 5.2 shows the QQ plots of every selecting algorithm and scenario combination that visually compare our data with the normal distribution. As said in [56], a K-S test is adequate to compare two continuous distributions when samples are small. Table 5.5 shows the respective D and p-Value of applying the K-S test to our outcomes, after data normalization, calculated as described in [58]. Since all p-Values are insignificant, we can reject the hypothesis that our data does not follow the normal distribution. In this condition, it is safe to apply the Student's t-test to compare the meanings from Scenarios 01 and 02.

5.5 Discussion

Table 5.6 shows the average cumulative reward for every algorithm-scenario combination, with the respective deviation. We calculated the estimated error using t-Student's distribution at 0.95 confidence. We proceed to compare cumulative reward using t-Value and p-Value when it is relevant to our experimental objectives. Table 5.7 shows the comparisons. Figure 5.6 presents the data from Table 5.6 overlaid by the exploration factor present in Table 5.4.

Table 5.4: Experimental outcomes over algorithm and scenario.

Selection algorithm	Scenario	Cumulative reward		Exploration factor		tc5 assignments	
		Average	SD	Average	SD	Average	SD
Random	01	721	69	0.76	0.015	39.67	6.23
Random	02	722	64	0.76	0.020	37.92	4.80
MAB UCB1	01	855	27	0.75	0.001	29.90	2.35
MAB UCB1	02	796	24	0.77	0.015	46.17	3.21
ReNoS	01	839	78	0.74	0.015	32.67	5.66
ReNoS	02	845	56	0.75	0.015	31.17	6.04
ReNoS-II	01	1,205	108	0.59	0.084	17.00	4.88
ReNoS-II	02	1,177	143	0.60	0.084	19.83	7.48
GT	01	1,558	34	0.41	0.100	7.75	2.73
GT	02	1,551	26	0.42	0.103	5.50	2.61

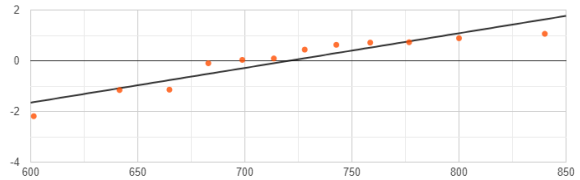
Table 5.5: Results of K-S test for normality in data from Table 5.4.

Algorithm - Scenario	D	p-Value
Random - Scenario 01	0.2111	0.1476
Random - Scenario 02	0.2178	0.1199
UCB1 - Scenario 01	0.1839	0.3170
UCB1 - Scenario 02	0.1380	0.7627
ReNoS - Scenario 01	0.1764	0.3273
ReNoS - Scenario 02	0.1769	0.3230
ReNoS-II - Scenario 01	0.1823	0.3302
ReNoS-II - Scenario 02	0.2058	0.1727
GT - Scenario 01	0.1532	0.6674
GT - Scenario 02	0.1545	0.6542

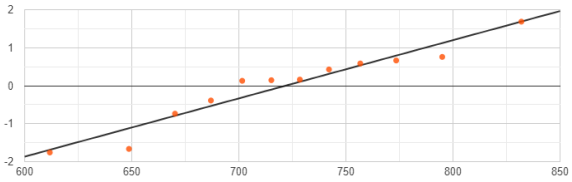
Table 5.6: Confidence interval for cumulative reward in Table 5.4.

Algorithm - Scenario	Average	SD	Estimated Error	Confidence interval at 95% confidence
Random - Scenario 01	721	69	151.30	569.04 to 872.49
Random - Scenario 02	722	64	140.10	581.91 to 861.92
UCB1 - Scenario 01	855	27	59.01	796.40 to 914.43
UCB1 - Scenario 02	796	24	52.42	743.40 to 848.24
ReNoS - Scenario 01	839	78	172.74	666.25 to 1,011.73
ReNoS - Scenario 02	845	56	123.07	721.81 to 967.94
ReNoS-II - Scenario 01	1,205	108	237.92	967.18 to 1,443.02
ReNoS-II - Scenario 02	1,177	143	315.54	861.79 to 1,492.87
GT - Scenario 01	1,508	34	73.76	1,483.76 to 1,631.28
GT - Scenario 02	1,551	26	56.76	1,493.94 to 1,607.47

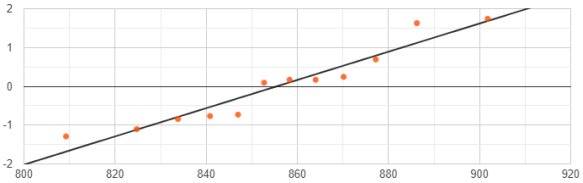
Observing cumulative reward measures in Table 5.4, we can see that when the broker applied random selection, the reward was worse than with other selection algorithms. It



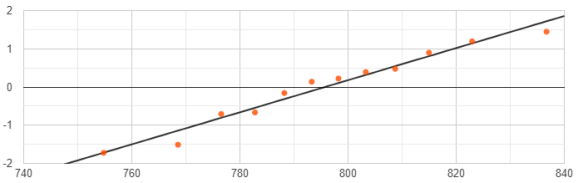
(a) Random in Scenario 01.



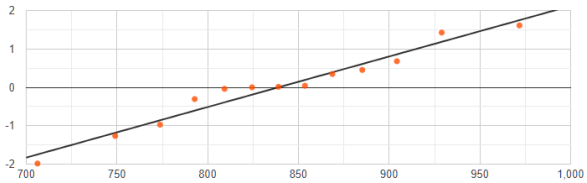
(b) Random in Scenario 02.



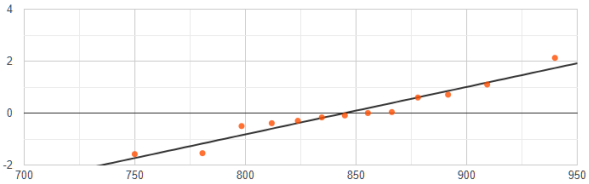
(c) UCB1 in Scenario 01.



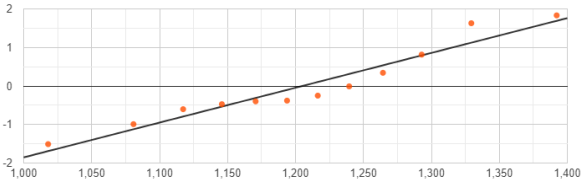
(d) UCB1 in Scenario 02.



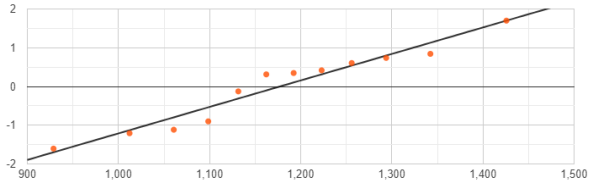
(e) ReNoS in Scenario 01.



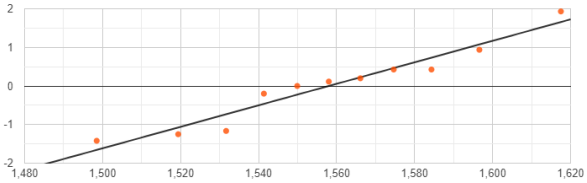
(f) ReNoS in Scenario 02.



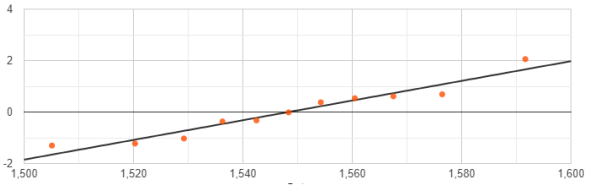
(g) ReNoS-II in Scenario 01.



(h) ReNoS-IIin Scenario 02.



(i) GT in Scenario 01.



(j) GT in Scenario 02.

Figure 5.2: QQ-Plot of outcomes over algorithms and scenarios.

indicates that learning about available performance transcoders and choosing those who could cope with the transcoding tasks makes a difference. Also, when the broker used random, there was no significant difference between Scenarios 01 and 02. Of course, the fake feedback strategy has no effect if the broker selects transcoders randomly.

When the broker applied MAB UCB1, the input was the average rating that viewers' proxies reported. Regarding Scenario 01, cumulative reward employing MAB UCB1 was significantly higher than that applying random, despite the exploration factor being closed for the two selection algorithms. It indicates that when there is a difference among expected returns from alternatives, the broker which applied MAB UCB1 was able to

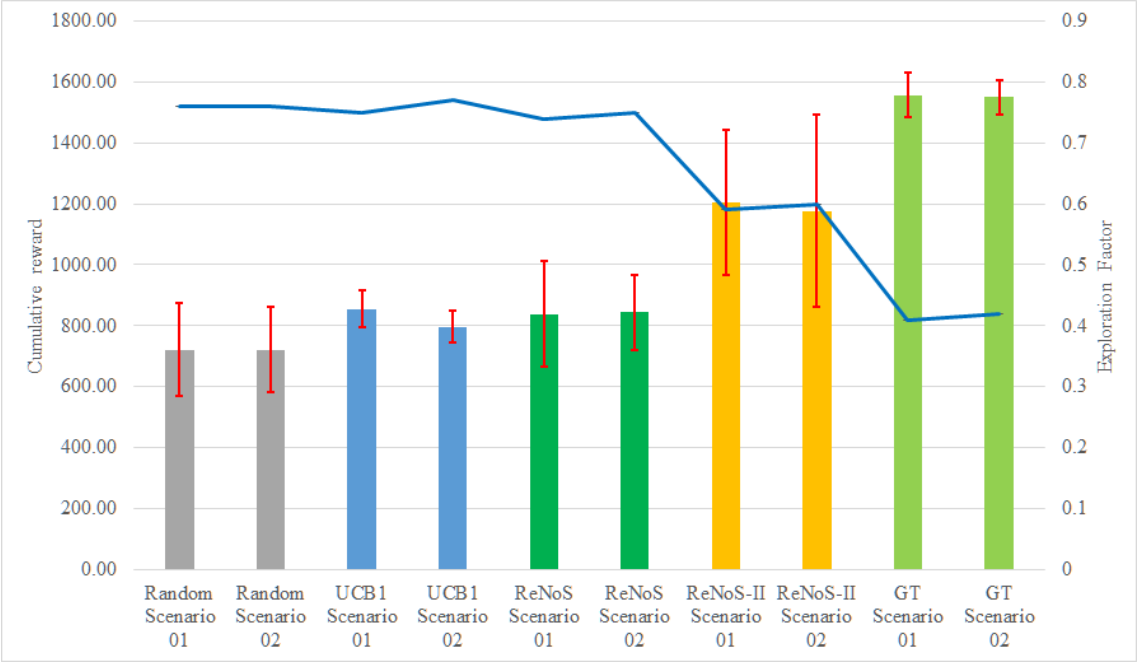


Figure 5.3: Broker's cumulative reward overlaid by exploration factor.

Table 5.7: Cumulative reward comparison using t-Test at 95% confidence.

Comparison	t-Value	p-Value	Interpretation
Random - Sc. 01 with 02	-0.400	≥ 0.05	No significant difference.
MAB UCB1 - Sc. 01 with 02	5.7563	< 0.05	Scr. 01 > Scr. 02.
ReNoS - Sc. 01 with 02	-0.2114	≥ 0.05	No significant difference.
ReNoS-II - Scr. 01 with 02	0.5357	≥ 0.05	No significant difference.
GT - Scr. 01 with 02	0.7342	≥ 0.05	No significant difference.
MAB UCB1 with Random in Scr. 01	6.3156	< 0.05	MAB UCB1 > Random.
MAB UCB1 with Random in Scr. 02	3.7748	< 0.05	MAB UCB1 > Random.
MAB UCB1 with ReNoS in Scr. 01	0.6860	≥ 0.05	No significant difference.
MAB UCB1 with ReNoS in Scr. 02	-2.7957	< 0.05	ReNoS > MAB UCB1.
ReNoS-II with ReNoS in Scr. 01	9.4939	< 0.05	ReNoS-II > ReNoS
ReNoS-II with ReNoS in Scr. 02	7.4842	< 0.05	ReNoS-II > ReNoS
GT with ReNoS-II in Scr. 01	10.3256	< 0.05	GT > ReNoS-II.
GT with ReNoS-II in Scr. 02	8.4393	< 0.05	GT > ReNoS-II.

explore alternatives and exploit the acquired knowledge to improve its choices. However, about Scenario 02, from Table 5.7 we can observe a significant decrease in cumulative reward from Scenario 01. Simultaneously, the number of tasks assigned to transcoder $tc5$ increased significantly. It indicates that the fake feedback strategy succeeded in manipulating the broker choices, and this exploitation of that vulnerability resulted in a loss to the broker and viewers', as expected from results presented in [24].

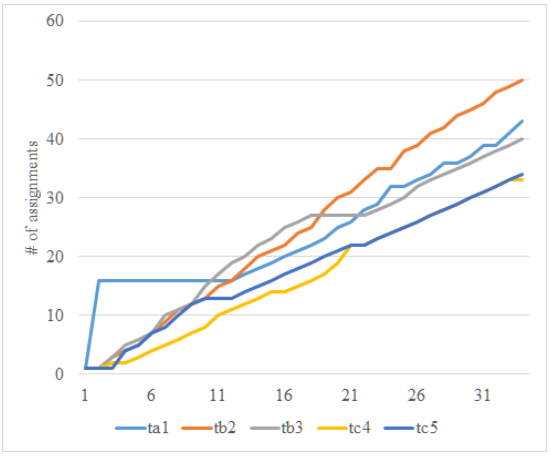
Proceeding to analyze the results obtained when the broker employed ReNoS in combination with FIRE T&RM, we see from the comparisons listed in Table 5.7 that cumulative

reward in Scenario 01 is comparable with that on MAB UCB1. We observe the same similarity in the exploration factor in Table 5.4. However, fake feedback attacks in Scenario 02 do not result in any damage to cumulative reward. As we also note, transcoder tc5 task assignments did not increase. These outcomes indicate that ReNoS combined with FIRE T&RM is a better selecting algorithm choice than MAB UCB1 when fake feedback attacks could occur.

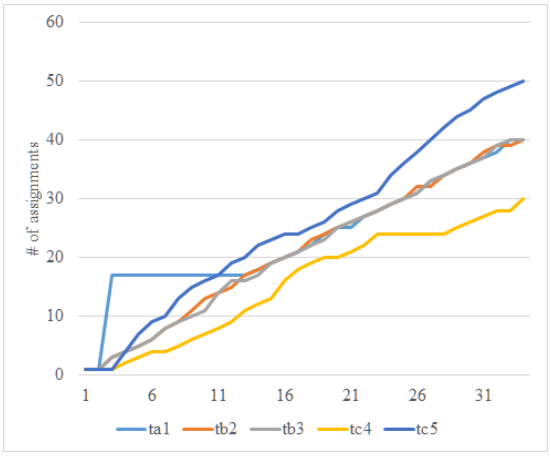
From Table 5.7, the broker's cumulative reward when ReNoS-II was employed is significantly higher than those obtained from MAB UCB1 and ReNoS in both scenarios, although the variance is similarly higher. Like with ReNoS, no significant decrease in cumulative reward exists between outcomes from Scenarios 01 and 02, which allows us to conclude that protection against fake feedback attacks is present. We can also note that the exploration factor is smaller for ReNoS-II than for MAB UCB1 and ReNoS. The exploration factor observed for ReNoS-II indicates that the highest evaluated transcoders received about 180% more tasks than any other transcoder available in the experiment.

When the broker employed the GT selecting algorithm, the data from Table 5.7 shows that cumulative reward is significantly higher than that observed for ReNoS-II, and, also, we can not observe harmful effects from fake feedback attacks in Scenario 02, accordingly with Table 5.4. However, GT's exploration factor was smaller than any other tested selecting algorithms. An exploration factor of only 0.41 means that the highest-rated transcoder received 59% of all transcoding tasks, or, from another perspective, about 5.86 times more than any other transcoder. There is a correlation between cumulative reward and the exploration factor since stochastic MAB policies tend to prioritize exploiting in the late turns. This correlation can be seen in Figure 5.3. However, if the exploration factor is too low, some transcoders might be overloaded, which can result in a small cumulative reward and higher regret.

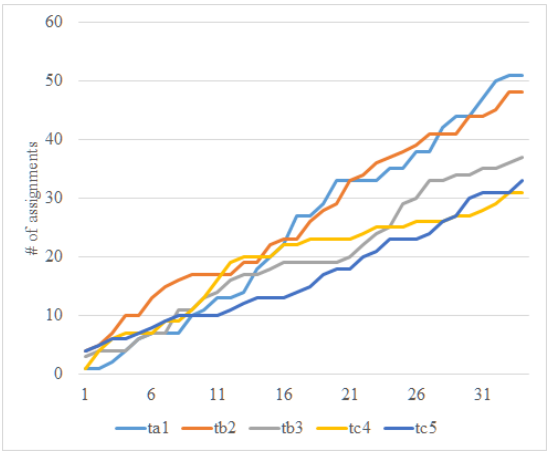
We can explain how the witnesses' credibility evaluation of FIRE T&RM reduced the damage of fake feedback attacks observing the distribution of transcoding tasks over time, which we represent in Figure 5.4. Figures 5.4a and 5.4b present the tasks assignments at one of the repetitions when MAB UCB1 was the selecting algorithm. We can see that in Scenario 2, the transcoder $tc5$ performed many more transcoding tasks, which increased its reward amount. However, since transcoders of C profile are not as good as transcoder of other profiles, the broker's final reward decreased. Therefore, we can conclude that the $tc5$ strategy on fake feedback in manipulating the broker's choices was successful. The same did not occur when ReNoS were guiding broker's choices. We can see that $tc5$ assignments decreased when we compare Figure 5.4c with 5.4d. Thus, we conclude that filtering the ratings employing the witnesses' credibility evaluation, as defined in the FIRE model, mitigated the effects of fake feedback.



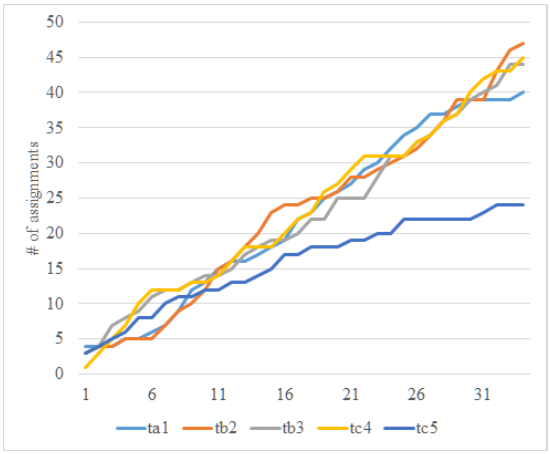
(a) UCB1 assignments in Scenario 1.



(b) UCB1 assignments in Scenario 2.



(c) ReNoS assignments in Scenario 1.



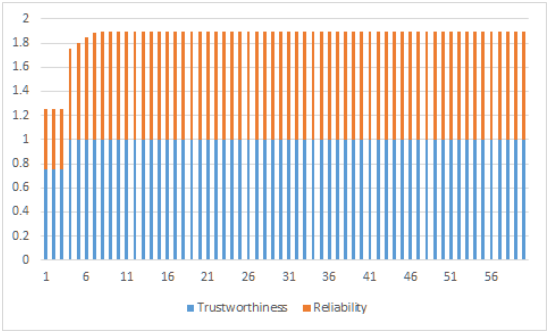
(d) ReNoS assignments in Scenario 2.

Figure 5.4: Transcoding assignments over algorithms and Scenarios 1 and 2.

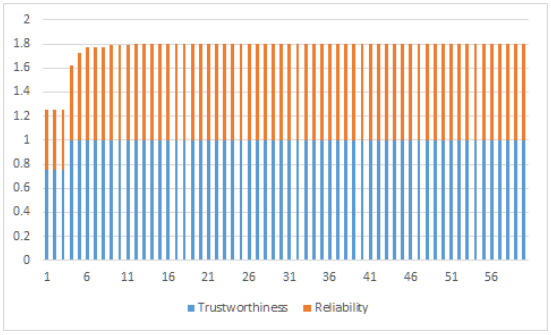
5.5.1 Analysing ReNoS-II

One of the differences between ReNoS and ReNoS-II is that ReNoS-II uses the reliability measure besides the trust and reputation value. A low reputation value can mean two things: that the agent does not present a higher performance, or the agent performance is volatile. These two characteristics are not desirable in a transcoder agent. Thus, brokers should defer transcoders with low-reliability values in favor of those with higher reliability.

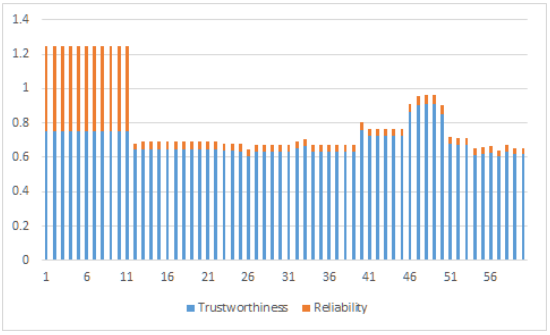
Figure 5.5 shows how trust and reputation, and reliability values evolved with ReNoS-II, meanwhile execution of the two scenarios. We observe that trust and reliability have a positive covariance in the FIRE model. Besides, for the *tc*5 agent, the reliability value increased in Scenario 2, although trust and reputation values remained almost the same in Figures 5.5c and 5.5d.



(a) Trust & reliability of $ta1$ in Scenario 1.



(b) Trust & reliability of $ta1$ in Scenario 2.



(c) Trust & reliability of $tc5$ in Scenario 1.

(d) Trust & reliability of $tc5$ in Scenario 2.

Figure 5.5: Evolution of trust and reliability over time with ReNoS-II.

5.5.2 Considerations about the exploration factor

We have observed that the exploration factor correlates with the cumulative reward. As can be seen in Figure 5.3, as lower is exploration, the higher is the broker's cumulative reward. Also, the difference between experimental scenarios does not seem to affect the exploration factor. However, we are afraid that analysis using this metric can not be extrapolated beyond the limits of this experiment. The first safeguard is that available choices were well behaved and remained fixed during all experimental repetitions. If options do not change, and there is an option that is better than all the others, selecting it will increase the cumulative reward when simultaneously decreasing the chance of others' selection. Another safeguard is that algorithms' users can tune parameters to adapt the balance between exploration and exploitation to the application scenario, although our experimental setup did not explore this possibility.

An empirical effort to characterize algorithms under the exploration factor should require a more dynamic setup and extrapolates the objectives of this dissertation. However, although the presented arguments, the exploration factor is still helpful to analyze our outcomes. Anyway, we consider that conclusions are safe if made under the other two measures.

5.6 Final considerations

Regarding the main objective of our experiments, the outcomes and the following analyses allow us to conclude that our approach performed as expected. The prototype could adequately coordinate the simulated distributed transcoding in conditions similar to those present in real-world situations. Considering the specific objectives:

- We conclude that combining our algorithms with T&RM evaluation improves node selection from comparing the scenarios with and without fake feedback. We observed that all tested algorithms performed significantly better than random choice. We also observed a significant decrease in the broker's cumulative reward when MAB UCB1 and simple averaging was applied. The same reduction did not occur when witnesses were first evaluated concerning their credibility using the FIRE T&RM.
- The broker's cumulative reward was higher when transcoder selection was guided by the algorithm ReNoS-II instead of ReNoS. Thus, we conclude that ReNoS-II outperforms ReNoS in conditions compatible with those in the conducted experiments.

Chapter 6

Conclusion

In this work, we presented a multi-agent architecture for dealing with the problem of distributing live video transcoding throughout FEC nodes and end-user devices. The architecture related three software agent roles with well-defined responsibilities, the viewer proxy, the transcoder, and the broker. The agents performing the broker role are responsible for coordinating the interaction of the other two. We also suggested where to place agents into the layers of the FEC network architecture to achieve the desired improvement on QoS. Since delegating tasks in such an open and dynamic environment as FEC can be risky, we explained how T&RM should be applied to evaluate the performance of transcoders as delegates and the credibility of the viewer proxies as witnesses. Then, we presented two algorithms for partner selection, ReNoS, and ReNoS-II, which take advantage of reputation reports from viewers to select the best nodes for performing the transcoding tasks.

We carried out experiments to validate the proposed architecture. We compared our algorithms with MAB UCB1 in two similar scenarios, one of those involving untrustworthy agents that tried to manipulate broker's choices. Analyzing the outcomes, we could see that ReNoS is as performative as UCB1, and ReNoS-II outperformed the other two in both scenarios.

MAB algorithms are designed to cope with the trade-off between exploring an unknown population of individuals and exploiting based on the pursuit of knowledge. The MAB algorithms depend on accumulating information in an evaluation function, which frequently is not much more elaborate than the simple average metric. Our experiments have shown that weighting and filtering information based on witness credibility can protect MAB stochastic algorithms from manipulative agents' behavior. Applying the FIRE model evaluations in our experiment, we could mitigate the effects of fake feedback attacks.

Applying the Tropos methodology [30], as described in Section 2.1.2, was essential to

understand the requirements from the point of view of means-end agents, allowing us to, as much as possible, preserve the autonomy of involved agents. The review of previews work that proposed architectures to distributed transcoding on FEC helped us identify the roles in common and where the agent approach could contribute to the discussion.

Regarding the research objectives (Chapter 1), we consider that the proposed multi-agent architecture combined the robustness of T&RM and stochastic and MAB algorithms to mitigate the risk of fake feedback attacks in open environments considering the context of real-time video transcoding on the FEC (Chapter 4). About the secondary objectives:

- The multi-agent architecture presents three agent roles with defined goals and actions. Performance measures such as the viewer's and broker's cumulative reward were used, as well as the playback assessment function (Chapter 4).
- The ReNoS and ReNoS-II algorithms presented in Section 4.5 and Chapter 4 successfully select partners to perform the transcoding tasks combining T&RM and MAB algorithms.
- Experimental outcome presented in Chapter 5 mainly indicates that our proposal is feasible for live-video distributed transcoding in open environments.

The described achievements of this work indicate the validity of our hypothesis that a multi-agent system could improve partnership on FEC, despite proper proof would only be achievable under a real-world application. As previews works, this is a step directly toward solving the problem of safely distributing tasks over a collection of unknown potential partners.

6.1 Publications

During this research project we have worked on the following articles:

- Charles A. N. Costa, Bruno Macchiavello, and Celia G. Ralha, Trust and reputation multiagent-driven model for distributed transcoding on fog-edge, in *Proceedings of the* 21^{st} *International Workshop on Trust in Agent Societies*, co-located with the 20^{th} *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 3-7 May 2021, Virtual. Available online at `http://ceur-ws.org/Vol-3022/paper5.pdf` [24].
- Charles A. N. Costa, and Celia G. Ralha, Reducing fake feedback damage to stochastic bandits algorithms with reputation, submitted to the *AI Communications* (The European Journal of Artificial Intelligence), IOS Press (under review).

6.2 Future work

We have left some open questions which worth consideration in future work. We do not suggest any reasoning model for the transcoder role, although the decisions made by transcoding agents substantially affect its utility. What should a transcoder consider when negotiating contracts with brokers? When would they give up the compensation refusing a transcoding job offer? Should transcoders be informed about their reputation? Besides, considering the services that CDN nodes deliver, an agent role specialized in caching services should be desirable.

Our architecture has similarities with recommendation systems regarding the selecting algorithms employed by the broker role. This fact instigates further research related to the cold-start problem. While ReNoS and ReNoS-II have mechanisms to allow new transcoders' selection with little or no information about them, further investigation is necessary. Questions like the following ones would be subject of interest. How fast a recently registered untrustworthy transcoder can affect a broker's selections? Is there a trade-off between avoiding the popularity bias and being vulnerable to fake-feedback attacks? Does the T&RM employed by the broker affect these issues? To answer such questions new investigation needs to be done.

References

[1] Buyya, Rajkumar and Satish N. Srirama: *Fog and Edge Computing*, chapter Internet of things (IoT) and new computing paradigms, pages 3–21. John Wiley & Sons, Inc., Hoboken, New Jersey, 2019. xiv, 21, 22

[2] Dabrowski, Marek, Robert Kolodynski, and Wojciech Zielinski: *Analysis of video delay in internet TV service over adaptive HTTP streaming.* In *Position Papers of the 2015 Federated Conf. on Computer Science and Information Systems*, pages 143–150, Lodz, Poland, oct 2015. PTI. xiv, 21, 23

[3] Chang, Zhi H., Bih F. Jong, Wei J. Wong, and M. L. Dennis Wong: *Distributed video transcoding on a heterogeneous computing platform.* In *2016 IEEE Asia Pacific Conf. on Circuits and Systems (APCCAS)*, pages 444–447, Jeju, Republic of Korea, oct 2016. IEEE. xiv, 2, 26, 27

[4] He, Qiyun, Cong Zhang, Xiaoqiang Ma, and Jiangchuan Liu: *Fog-based transcoding for crowdsourced video livecast.* IEEE Communications Magazine, 55(4):28–33, apr 2017. xiv, 2, 3, 4, 24, 26, 27, 28, 29

[5] Liu, Xingchi, Mahsa Derakhshani, and Sangarapillai Lambotharan: *Joint transcoding task assignment and association control for fog-assisted crowdsourced live streaming.* IEEE Communications Letters, 23(11):2036–2040, nov 2019. xiv, 2, 3, 24, 26, 28, 29

[6] Bilal, Kashif, Emna Baccour, Aiman Erbad, Amr Mohamed, and Mohsen Guizani: *Collaborative joint caching and transcoding in mobile edge networks.* Journal of Network and Computer Applications, 136:86–99, 2019, ISSN 1084-8045. xiv, 2, 3, 24, 26, 29, 30

[7] Fu, F., Y. Kang, Z. Zhang, and F. R. Yu: *Transcoding for live streaming-based on vehicular fog computing: An actor-critic drl approach.* In *IEEE Conference on Computer Communications Workshops (INFOCOM)*, pages 1015–1020, Toronto, ON, Canada, 2020. IEEE. xiv, 2, 3, 30, 31

[8] Wang, Fangxin, Jiangchuan Liu, Cong Zhang, Lifeng Sun, and Kai Hwang: *Intelligent edge learning for personalized crowdsourced livecast: Challenges, opportunities, and solutions.* Netwrk. Mag. of Global Internetwkg., 35(1):170–176, mar 2021, ISSN 0890-8044. `https://doi.org/10.1109/MNET.011.2000281`. xiv, 2, 24, 26, 31, 32

[9] Chen, Xingyan, Changqiao Xu, Mu Wang, Zhonghui Wu, Lujie Zhong, and Luigi Alfredo Grieco: *Augmented queue-based transmission and transcoding optimization*

for livecast services based on cloud-edge-crowd integration. IEEE Transactions on Circuits and Systems for Video Technology, 31(11):4470–4484, 2021. xiv, 2, 24, 26, 32, 33

[10] Wooldridge, Michael: *An introduction to multiagent systems.* Wiley, Chichester, UK, 2nd edition, 2009, ISBN 978-0-470-51946-2. 1, 6, 7

[11] Oprea, Mihaela: *Applications of multi-agent systems.* In *Information Technology*, pages 239–270. Kluwer Academic Publishers, 2004. 1

[12] Cisco: *Cisco visual networking index: Forecast and trends, 2017-2022.* Technical report, Cisco, 2017. 1

[13] Duanmu, Zhengfang, Kede Ma, and Zhou Wang: *Quality-of-experience for adaptive streaming videos: An expectation confirmation theory motivated approach.* IEEE Transactions on Image Processing, 27(12):6135–6146, dec 2018. 2, 23

[14] Spiteri, Kevin, Rahul Urgaonkar, and Ramesh K. Sitaraman: *BOLA: Near-optimal bitrate adaptation for online videos.* In *The 35th Annual IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1–9, San Francisco, CA, apr 2016. IEEE. 2, 23, 51

[15] Bing, Benny: *Next-generation video coding and streaming.* John Wiley & Sons, Inc, Hoboken, New Jersey, sep 2015. 2

[16] 23009-1:2012, ISO/IEC: *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats.* Standard, Int. Organization for Standardization, Geneva, CH, 2012. 2

[17] Pires, Karine and Gwendal Simon: *DASH in twitch.* In *Proc. of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, pages 13–18, Sydney, Australia, 2014. ACM Press. 2

[18] Slivkins, Aleksandrs: *Introduction to multi-armed bandits.* Foundations and Trends in Machine Learning, 12(1-2):1–286, 2019. 2, 18, 20

[19] Liu, L. and W. Shi: *Trust and reputation management.* IEEE Internet Computing, 14(5):10–13, 2010. 2, 17

[20] Granatyr, Jones, Vanderson Botelho, Otto R. Lessing, Edson E. Scalabrin, Jean Paul Barthès, and Fabrício Enembreck: *Trust and reputation models for multiagent systems.* ACM Computing Surveys, 48(2):1–42, nov 2015. 3

[21] Vallée, Thibaut, Grégory Bonnet, and François Bourdon: *Multi-armed bandit policies for reputation systems.* In Y., Demazeau, Zambonelli F., Corchado J.M., and Bajo J. (editors): *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, volume 8473 of *Lecture Notes in Computer Science*. Springer, Cham, 2014. 3, 21

[22] Huang, Hongbing, Guiming Zhu, and Shiyao Jin: *Revisiting trust and reputation in multi-agent systems.* In *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, pages 424–429, Guangzhou, People's Republic of China, 2008. IEEE. 3

[23] Huynh, Trung D., Nicholas R. Jennings, and Nigel R. Shadbolt: *An integrated trust and reputation model for open multi-agent systems.* Autonomous Agents and Multi-Agent Systems, 13(2):119–154, mar 2006. 3, 12, 14, 17, 56, 62

[24] Costa, Charles A. N., Bruno Macchiavello, and Célia G. Ralha: *Trust and reputation multiagent-driven model for distributed transcoding on fog-edge.* In *Proc. of 21st International Workshop on Trust in Agent Societies*, 2021. `http://ceur-ws.org/Vol-3022/paper5.pdf`. 4, 50, 56, 59, 60, 62, 68, 74

[25] Russell, Stuart and Peter Norvig: *Artificial intelligence: A modern approach.* Prentice Hall, 3rd edition, 2010. 6, 22, 51

[26] Yoav Shoham, Kevin Leyton Brown: *Multiagent systems.* Cambridge University Press, 2014, ISBN 0521899435. 6

[27] Lam, Ka man and Ho fung Leung: *A trust/honesty model in multiagent semi-competitive environments.* In *Intelligent Agents and Multi-Agent Systems*, pages 128–147. Springer Berlin Heidelberg, 2005. 6

[28] Weiss, Gerhard: *Multiagent systems: a modern approach to distributed artificial intelligence.* MIT Press, Cambridge, Mass, 1999, ISBN 0262232030. 7, 8

[29] Sichman, Jaime S. and Helder Coelho: *Autonomous agents and multi-agent systems.* In Lopes, Fernando and Helder Coelho (editors): *Negotiation and argumentation in multi-agent systems*, pages 3–29. Bentham Science Publishers, apr 2014. 8

[30] Bresciani, Paolo, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos: *Tropos: An agent-oriented software development methodology.* Autonomous Agents and Multi-Agent Systems, 8(3):203–236, may 2004. 9, 10, 11, 73

[31] Dalpiaz, Fabiano, Xavier Franch, and Jennifer Horkoff: *istar 2.0 language guide*, 2016. `https://arxiv.org/pdf/1605.07767.pdf`, Accessed: 2022-01-23. 9, 11

[32] Pimentel, João and Jaelson Castro: *pistar tool – a pluggable online tool for goal modeling.* In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 498–499, 2018. 9

[33] Castelfranchi, C. and R. Falcone: *Principles of trust for MAS: cognitive anatomy, social importance, and quantification.* In *Proc. Int. Conf. on Multi Agent Systems*, pages 72–79. IEEE Comput. Soc, 1998. 12, 21

[34] Huynh, T. Dong, Nicholas R. Jennings, and Nigel R. Shadbolt: *On handling inaccurate witness reports.* In *Proc. of 8th Int. Workshop on Trust in Agent Societies*, pages 63–77, 2005. `http://eprints.soton.ac.uk/261136/`. 12, 15, 16, 56, 62

[35] Marsh, Stephen: *Formalising Trust as a Computational Concept.* PhD thesis, University of Stirling, July 1999. 12, 13, 57

[36] Sabater-Mir, Jordi and Carles Sierra: *REGRET.* In *Proc. of the* 5^{th} *Int. Conf. on Autonomous agents.* ACM Press, 2001. 12

[37] Zacharia, Giorgos and Pattie Maes: *Trust management through reputation mechanisms.* Applied Artificial Intelligence, 14(9):881–907, oct 2000. 12, 13, 14

[38] Hoelz, Bruno W. P.: *Metamodelo para adaptação de confiança e reputação em sistemas multiagente dinâmicos.* PhD thesis, Universidade de Brasília, September 2013. 12

[39] Hoelz, Bruno W. P. and Célia G. Ralha: *Towards a cognitive meta-model for adaptive trust and reputation in open multi-agent systems.* Auton. Agents Multi Agent Syst., 29(6):1125–1156, 2015. `https://doi.org/10.1007/s10458-014-9278-9`. 12

[40] Hoelz, Bruno W. P. and Célia G. Ralha: *Towards a cognitive meta-model for adaptive trust and reputation in open multi-agent systems.* In Jonker, Catholijn M., Stacy Marsella, John Thangarajah, and Karl Tuyls (editors): *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, pages 624–625. ACM, 2016. 12

[41] Zhang, Yu, Jing Bian, and Weixiang Zhu: *Trust fraud: A crucial challenge for china's e-commerce market.* Electronic Commerce Research and Applications, 12(5):299–308, 2013, ISSN 1567-4223. Chinese E-Commerce. 17, 60

[42] Das, Anupam and Mohammad M. Islam: *SecuredTrust: A dynamic trust computation model for secured communication in multiagent systems.* IEEE Transactions on Dependable and Secure Computing, 9(2):261–274, 2012. 17

[43] Teacy, W. T. Luke, Jigar Patel, Nicholas R. Jennings, and Michael Luck: *TRAVOS: Trust and reputation in the context of inaccurate information sources.* Autonomous Agents and Multi-Agent Systems, 12(2):183–198, 2006. 17

[44] Noor, Talal H., Quan Z. Sheng, Abdullah Alfazi, Jeriel Law, and Anne H. H. Ngu: *Identifying fake feedback for effective trust management in cloud environments.* In Ghose, Aditya, Huibiao Zhu, Qi Yu, Alex Delis, Quang Z. Sheng, Olivier Perrin, Jianmin Wang, and Yan Wang (editors): *Service-Oriented Computing - ICSOC 2012 Workshops*, pages 47–58, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg, ISBN 978-3-642-37804-1. 17

[45] Siadat, Safieh, Amir M. Rahmani, and Hamidreza Navid: *Identifying fake feedback in cloud trust management systems using feedback evaluation component and bayesian game model.* J. Supercomput., 73(6):2682–2704, jan 2017. 17

[46] Vermorel, Joannès and Mehryar Mohri: *Multi-armed bandit algorithms and empirical evaluation.* In *Proceedings of the 16th European Conference on Machine Learning*, ECML'05, page 437–448, Berlin, Heidelberg, 2005. Springer-Verlag, ISBN 3540292438. 18, 19

[47] Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer: *Finite-time analysis of the multiarmed bandit problem.* Machine Learning, 47(2/3):235–256, 2002. 20

[48] Pandey, Sandeep, Deepak Agarwal, Deepayan Chakrabarti, and Vanja Josifovski: *Bandits for taxonomies: A model-based approach.* In *Proceedings of the 2007 SIAM International Conference on Data Mining.* Society for Industrial and Applied Mathematics, apr 2007. 20

[49] Auer, Peter, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire: *Gambling in a rigged casino: The adversarial multi-armed bandit problem.* In *Proceedings of IEEE 36th Annual Foundations of Computer Science.* IEEE Comput. Soc. Press, 1995. 20

[50] Lykouris, Thodoris, Vahab Mirrokni, and Renato P. Leme: *Stochastic bandits robust to adversarial corruptions.* In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing.* ACM, 2018. 20, 21

[51] Karagkioules, Theodoros, Cyril Concolato, Dimitrios Tsilimantos, and Stefan Valentin: *A comparative case study of HTTP adaptive streaming algorithms in mobile networks.* In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video - NOSSDAV'17.* ACM Press, 2017. 23

[52] Wang, Cong, Divyashri Bhat, Amr Rizk, and Michael Zink: *Design and analysis of QoE-aware quality adaptation for DASH.* ACM Transactions on Multimedia Computing, Communications, and Applications, 13(3s):1–24, aug 2017. 23

[53] Li, Zhenhua, Yan Huang, Gang Liu, Fuchen Wang, Zhi Li Zhang, and Yafei Dai: *Cloud transcoder.* In *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video - NOSSDAV '12.* ACM Press, 2012. 24

[54] Adhikari, Vijay K., Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi Li Zhang: *Unreeling netflix: Understanding and improving multi-CDN movie delivery.* In *2012 Proceedings IEEE INFOCOM.* IEEE, mar 2012. 24

[55] Liu, Dongyu, Eric Setton, Bo Shen, and Songqing Chen: *Pat: Peer-assisted transcoding for overlay streaming to heterogeneous devices.* In *Proc. Int. Workshop Netw. Oper. Syst. Support Dig. Audio Video*, 2007. 24

[56] Raj, Jain: *The art of computer systems performance analysis.* WILEY, New York, 1991, ISBN 0471503363. 24, 25, 65

[57] Spiegel, Murray: *Schaum's outline of theory and problems of statistics.* McGraw-Hill, New York, 2008, ISBN 0071594469. 25

[58] Verzani, John: *Using R for introductory statistics.* Chapman & Hall/CRC, Boca Raton, 2005, ISBN 1584884509. 25, 65

[59] Hilvert-Bruce, Zorah, James T. Neill, Max Sjöblom, and Juho Hamari: *Social motivations of live-streaming viewer engagement on twitch.* Computers in Human Behavior, 84:58–67, jul 2018. 36

[60] OMG: *OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1*, August 2011. http://www.omg.org/spec/UML/2.4.1. 47

[61] Thompson, William R.: *On the likelihood that one unknown probability exceeds another in view of the evidence of two samples*. Biometrika, 25(3-4):285–294, December 1933, ISSN 0006-3444. https://doi.org/10.1093/biomet/25.3-4.285. 55, 57

[62] Bellifemine, Fabio, Giovanni Caire, and Dominic Greenwood: *Developing multi-agent systems with JADE*. John Wiley & Sons, Ltd, Chichester, West Sussex, England, mar 2007. 60

[63] Costa, Charles A. N.: *DisTran – distributed transcoding simulator implemented in Java with JADE framework*. https://github.com/charlesANC/distran, 2021. 60