table of contents

[table of contents 1](#_Toc160711241)

Chapter1. [P2P Content Distribution: the view from an edge network 2](#_Toc160711242)

[*Methodology* 3](#_Toc160711243)

[*Hit* *Ratios* 5](#_Toc160711244)

# **Chapter 1. P2P Content Distribution: the view from an edge network**

In this section, we first look into the overhead of content distribution through BitTorrent as experienced by an edge network. We then quantify the savings that could be gained in the same scenario if locality was exploited. We conclude by examining the implications of locality-aware mechanisms on the user experience.

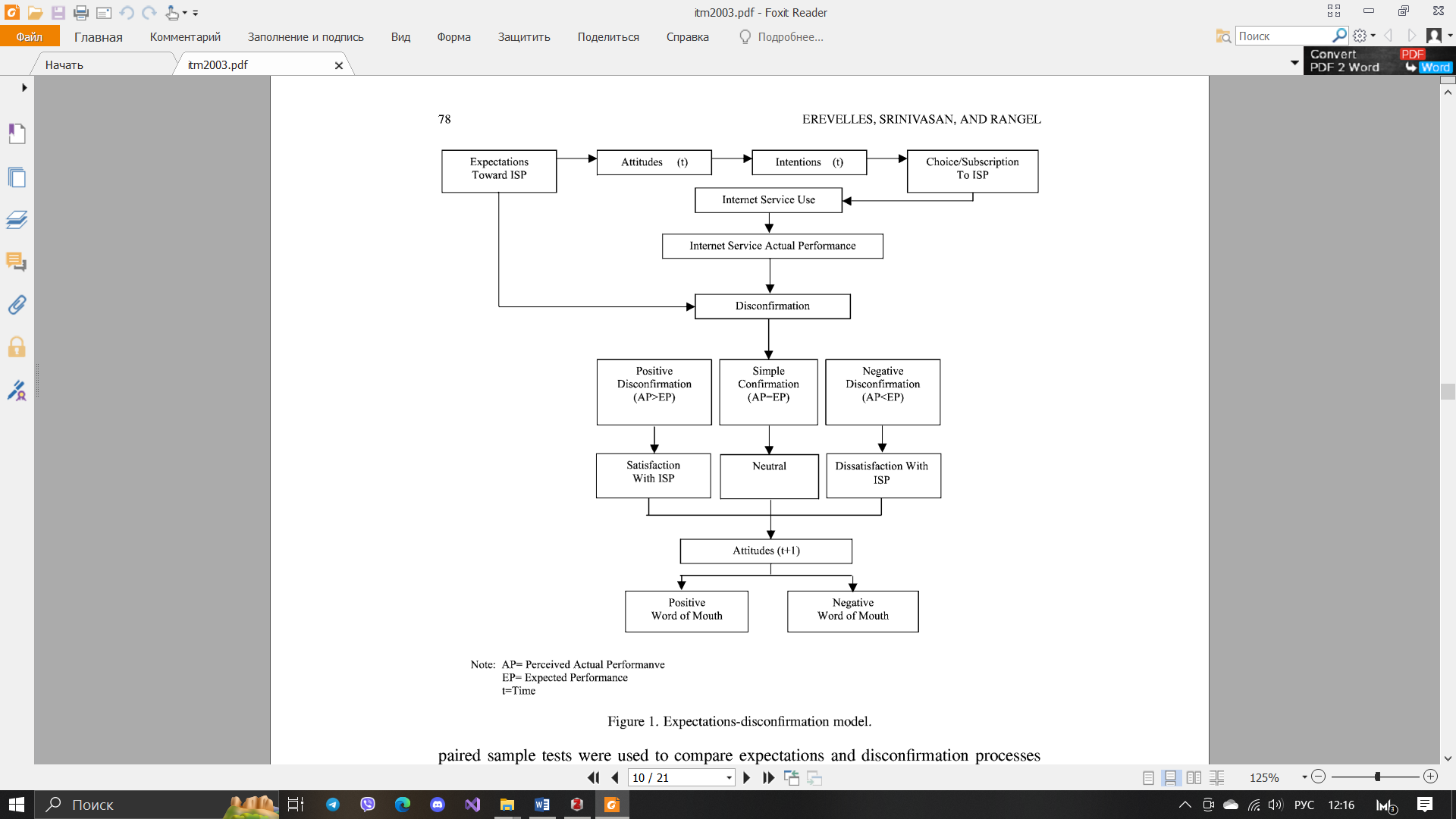


Fig. 1.1. Expectations-disconfirmation model

For this study we use three day-long packet traces collected with the high speed monitoring box described in (Table 1). The monitor is installed on the access-link (fullduplex Gigabit Ethernet link) of a residential university that hosts numerous academic, research, and residential complexes with an approximate population of 20,000 users (a population equivalent of a small ISP). Our monitors capture the TCP/IP header and adequate payload information from all packets crossing the link in both directions to enable the identification of specific applications. P2P traffic accounts for approximately a third of the traffic in each one of the three traces (identified by the methodology described in ), while 13%-15% of the total traffic (8%-11% of the total packets) in the link is due to BitTorrent flows. The large fraction of BitTorrent traffic reveals its growing popularity and offers a sufficient sample to study its dynamics.

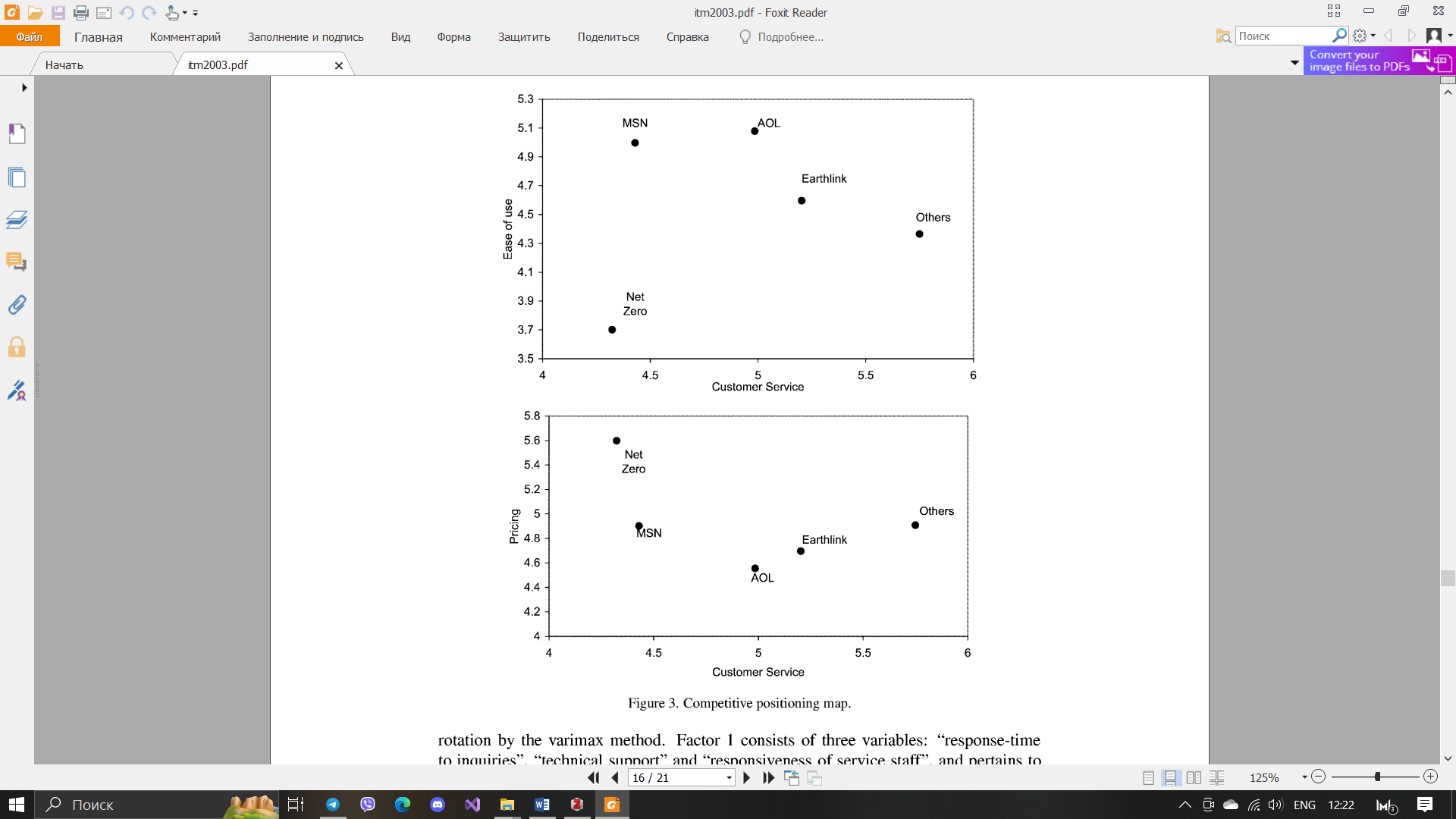


Fig. 1.2. Competitive positioning map

## **Methodology**

Studying the dynamics of the BitTorrent network in the traces involves:

1. identifying all BitTorrent packets;
2. determining their specific format;
3. reconstructing all interactions across all BitTorrent flows;

To identify BitTorrent flows and messages, we have developed a BitTorrent protocol dissector. While the Ethereal protocol analyzer [6] has a BitTorrent module, we encountered several problems with missed packets (e.g., TCP/IP packets that contained BitTorrent messages in the payload but were not identified by Ethereal), in the handling of out-of-order and fragmented BitTorrent messages and with multiple messages in the same TCP/IP packet. Furthermore, Ethereal’s BitTorrent module cannot dissect tracker HTTP request/responses. Specifically, we need to identify two types of messages for all BitTorrent flows:

1. the tracker request/responses;
2. the messages between peers.

The nominal format for all packets can be found in [28]. The tracker request consists -among other things- of the peer id, the file hash and the local IP of the BitTorrent client (optional).

Calculation of the information flow consumed by a group of subscribers:

. (1.1)

The tracker response is usually a list of available clients for the requested file with statistics regarding the number of seeds, leechers etc. Regarding peer interactions, detecting the following BitTorrent messages is crucial to our analysis:

* Handshake: The first BitTorrent message transmitted by the initiator of the connection. It specifies the hash of the file and the peer id.
* Piece: BitTorrent files are divided into Pieces. The size of each piece is usually 262KB − 1MB and pieces are further subdivided in blocks of typically 16KB. Blocks constitute the byte-segments that are actually transferred. The Piece message contains a data block of a given piece. The first nine bytes of the message specify the piece index, the byte offset within the piece and the size of the block.
* BitField: BitField specifies which pieces of the file are available for upload. It is a sequence of bits where set bits correspond to available pieces. It is typically the first message after the handshake.

Have: The Have message advertises that the sender has downloaded a piece and the piece is now available for upload.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set | Date | Day | Start | Dur | Direc. | Src.IP | Dst.IP | Packets | Bytes | Aver. Flows/  5min. |
| Jan | 2004-01-20 | Tue | 16:50 | 24.6 h | Bi-dir. | 2709 K | 2626 K | 2308 M | 1223 G | 596 K |
| Apr | 2004-04-23 | Fri | 15:40 | 33.6 h | Bi-dir. | 4502 K | 5742 K | 3402 M | 1652 G | 570 K |
| May | 2004-05-19 | Wed | 07:50 | 28.6 h | Bi-dir, | 1246 K | 1301 K | 3073 M | 1706 G | 799 K |

## **Hit Ratios**

We quantify locality in terms of hit ratios. The hit ratio (analogous to caching hit ratio) refers to content that has already been downloaded and is present locally within our monitored ISP.

We examine the hit ratio along three dimensions: **File hit ratio**, where we assume that the complete file is “cached” locally after the first download. Local caching would be the equivalent of a local-aware P2P system, where once a full copy exists within the ISP (either one peer has the full copy, or pieces of the file are spread across the ISP’s customers), requests are served locally (assuming always active peers). We determine the total network traffic using the formula:

, (1.2)

where, = (0.050.07)n ­– is the coefficient of service, broadcast and unaccounted traffic,

= (1,22,0) – margin factor to account for the future development of the network,

n – is the number of computers on the network.

Thus, the file hit ratio reveals the fraction of multiple downloads of the same content for the ISP in terms of the total number of downloaded files. Let Ni, be the user population for file i within the ISP with n being the total number of files. Then, the file hit ratio is defined as follows:

. (1.3)