

join request. Since the set of landmarks is not the same for all nodes anymore, each node obtains its coordinate vector  $y$  with respect to its landmark set  $\{L_j\}_{1 \leq j \leq n}$ . If two nodes find as basisvectors the set  $a_1, a_2, \dots, a_n$  and  $a'_1, a'_2, \dots, a'_n$  respectively, then any vector  $a_k$  can be expressed as a linear combination of the basisvectors  $a'_1, a'_2, \dots, a'_n$ . Hence, there exists a unique transformation matrix  $P$  such that  $A = PA'$  and  $P = A(A')^{-1}$ . Each node can transform its “local” coordinates  $y$  to “global” coordinates  $y' = P^{-1}y$  provided it knows the matrix  $A'$ .

Thus, in a scheme where the set of landmarks is dynamic, the entry node supplies the joining node with the “global” set of basis vectors and a set of peers from which landmarks can be chosen.

### 13.4.2.3 Embedding

By conceiving the network distances as the components of a vector, the node  $X_j$  is embedded in a  $D$ -dimensional hyperspace where  $D < n$ . The basic assumption of the landmark method is that the distance between nodes in the hyperspace, which can be computed from their coordinate vectors, is indicative of their network distance. Practice has shown that the method can indeed work.

Let  $d(A, B) \geq 0$  be the network distance<sup>7</sup> between node  $A$  and  $B$ . To each node  $X_i$ , we assign a coordinate vector

$$\mathbf{x}_i = (d(X_i, L_1), d(X_i, L_2), \dots, d(X_i, L_n))$$

For the distance between the coordinate vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , we use the notation  $\bar{d}(\mathbf{x}_i, \mathbf{x}_j)$ , which we call the “hyperspace metric” as opposed to the “network metric”. The central question of the landmark method is: To which extent can the hyperspace distance  $\bar{d}(\mathbf{x}_i, \mathbf{x}_j)$  be used to predict the network distance  $d(X_i, X_j)$ ?

## 13.5 BitTorrent

The BitTorrent protocol<sup>8</sup> is an open source peer-to-peer protocol used for an efficient distribution of digital records or files among peers. In *normal or classical downloading*, for example, when a file is downloaded after a user click on a hyperlink of a webpage, the http protocol sends the complete file from a server to the requesting host. As soon as a large number  $m$  of users are downloading the same file at the same time, the server has to upload  $m$  times the same file concurrently. In

<sup>7</sup> The network distance does not necessarily obey the three mathematical conditions that define a distance in a vector space: (a)  $d(A, B) = 0$  if and only if  $A = B$ . (b)  $d(A, B) = d(B, A)$  and (c) the triangle inequality,  $d(A, B) \leq d(A, C) + d(C, B)$ . In particular, as shown in van Langen *et al.* (2004), the triangle inequalities do not necessarily hold.

<sup>8</sup> The inventor of the BitTorrent protocol is Bram Cohen, whose description (Cohen, 2003) is followed here, although various updates and corrections of the BitTorrent protocol exist. The official website <http://www.bittorrent.org/developer.html> contains further detailed information. The BitTorrent protocol is used as basic transfer protocol in other peer-to-peer protocols, such as e.g. Tribler developed at TUDelft.

peer-to-peer downloading via BitTorrent, on the other hand, the cost of uploading is distributed more evenly over the  $m$  users and the server, since these  $m$  users also upload parts of the already downloaded file to each other.

This basic principle illustrates the power of peer-to-peer technology in which users also actively help to distribute information to other users within the network. This contrasts central distribution, where a large server assists each user individually and controls the transfer of content.

P2P technology has potential in a wide range of applications such as file sharing, distribution of TV or music, video on demand, network gaming, distributed storing, computing, and decentralized search engines. When real-time TV as news, sport events, etc. are broadcast to a million of end-users via normal downloading over the Internet, a large cluster of powerful servers (centrally coordinated) is needed. P2P technology seems a promising alternative.

### 13.5.1 Torrent file and tracker

Fig. 13.9 shows the actors in BitTorrent whose interrelation will be explained in this section.

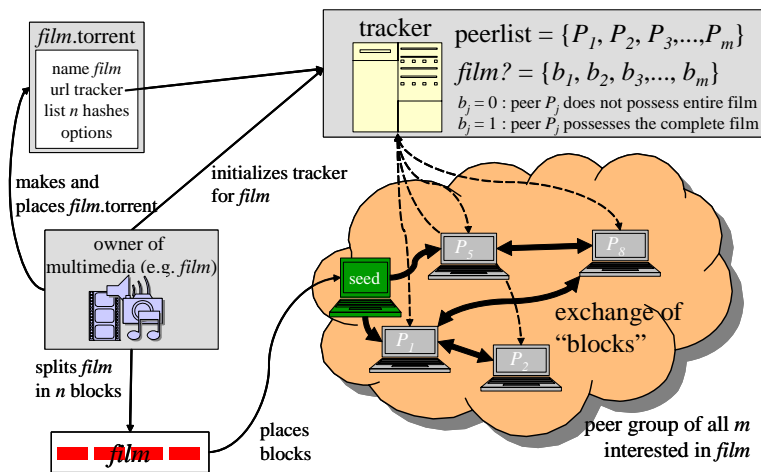


Fig. 13.9. Schematic overview of the actors in BitTorrent.

Before starting the BitTorrent downloading process of content, named *film* in Fig. 13.9, a static file with the extension \*.torrent must be available, e.g. on a web server. This torrent file contains the following information about the content: the length (number of bits) and the name of the file, hashing information (a key or code) and the URL of the "tracker". Thus, to produce a \*.torrent file, a complete record of the content is needed in order to be able to construct the requested hash

information. While the \*.torrent file is mostly placed on a public website, the actual content is usually on a different computer as will explained below.

*Trackers* are simple protocols operating on top of http that help downloaders of the same file to find each other. A downloader sends the filename he intends to download and the TCP port numbers (see Section 5.1). The tracker adds the downloader to the peerlist belonging to this \*.torrent file and returns a sublist of peers that are downloading the same file. This sublist is selected from a complete list via a certain algorithm: often a random number of  $k < m$  peers are selected from the complete list of  $m$  peers. The tracker thus stores only the complete list of peers  $\{P_1, P_2, \dots, P_m\}$  and a corresponding list  $\{b_1, b_2, \dots, b_m\}$  with binary information: for each  $1 \leq j \leq m$ , the bit  $b_j = 1$  (or  $b_j = 0$ ) denotes that the peer  $j$  has (or does not have) a complete copy of the file. As illustrated in Fig. 13.9, downloaders are using this sublist of peers to contact each other. During the downloading process, downloaders can contact the tracker again to obtain a new (sub)list that helps to improve and update their view on the peer group.

To be successful, at least one downloader with the complete file, called “seed”, should start. This means that the first seed only uploads and initiates the process. This first seed is usually the one that placed the \*.torrent and that wishes to distribute the corresponding content (such as *film* in Fig. 13.9) via BitTorrent. After having started the distribution, this seed can withdraw from the peer group, e.g. in order to avoid localization or to shield his identity, especially in illegal content distribution.

### 13.5.2 Downloading and uploading

BitTorrent separates a record of the content into several pieces or blocks with a certain size<sup>9</sup> (typically  $\frac{1}{4}$  Mbyte). For reasons of integrity as explained in Chapter 3, the pieces/blocks are given a hash code. The hash code of each block is contained in the \*.torrent file. When a file is split into  $n$  blocks, as in Fig. 13.9, the \*.torrent file contains a list of  $n$  hash codes. After executing a hash test, which is a simple integrity test based on the same principle as that of error detection (see Section 3.2), each downloader in the peer group reports to all other peers which blocks he already owns. Peers continuously try to download the missing blocks from other peer group members, while they upload their blocks to others. A major source of complexity of the BitTorrent protocol lies in the distributed exchange of information blocks between downloaders and uploaders.

During the distributed exchange process, BitTorrent employs (a) a selection strategy that defines which blocks should be downloaded first and (b) a fair mecha-

<sup>9</sup> Since BitTorrent uses TCP, the blocks of 1/4 Mbyte are further divided by TCP in smaller sizes (typically 16 kbyte) to optimize the TCP data transfer among peers.

nism between the uploading and downloading speed of every peer. It is important to select the blocks cleverly to optimize the exchange of blocks among the peers. BitTorrent prescribes that a block (that will be divided in sub-blocks by TCP) is downloaded completely before another block is downloaded. Furthermore, BitTorrent uses the “rarest first” strategy. This strategy aims to divide the already downloaded blocks over the peers as uniformly as possible. It is not difficult to see that, when all peers own the same blocks, no mutual uploading is needed, while efficiency in mutual uploading is gained when maximum exchange is possible. Therefore, the “rarest first” strategy makes downloaders to download different blocks from a seed in order to start the uploading process to peers as soon as possible.

### 13.5.3 Incentive mechanism for uploading

In order to distribute the blocks fairly and efficiently, peers need to be triggered, beside downloading (which is their final aim), also to participate in uploading. Each peer will maximize his downloading and can determine to which peer he will upload via a variant<sup>10</sup> of “tit-for-tat”. The peer chooses his own “tit-for-tat” algorithm. Basically, the peer first donates a block hoping to receive an interesting block from the other peer. Each peer uploads to that group of peers that offered it the highest downloading rate. Recent research studies variants of the “tit-for-tat”, that are optimal, simple, reliable and safe and that accelerate the downloading process e.g. by employing network coding (see Section A.7).

A peer can decide not to upload, which is named “choking”. The choking algorithm is strictly spoken not a part of BitTorrent, but crucial to equally divide the capacity. A peer that only downloads can expect to be choked by other peers such that his access to new blocks decreases. The cooperative “tit-for-tat” enforces that each peer needs to upload in order to download the complete file sufficiently fast. The “tit-for-tat” principle limits “free-riding” (downloading only) as much as possible. The “tit-for-tat” mechanism further leads to a Pareto optimum<sup>11</sup>.

It remains difficult for a peer to find an optimal strategy at any time to choose its best upload peers. The peers that possess the most different blocks and from whom downloading proceeds fastest, may seem most suitable. But the dynamic character of a peer group, in which peers can leave and join the group at any moment, also requires a dynamic upload and download selection strategy. For example, a peer 1 from which can be downloaded at high rate, may change its behavior or a new

<sup>10</sup> “tit-for-tat” is a social mechanism of collaboration. It was already known by the Romans as “do ut des”, which means “I give in order that you give”.

<sup>11</sup> A Pareto optimum in an interactive game is a position in which individual players cannot improve anymore. In other words, any other strategy that a player may choose will be counteracted by the other players. Hence, a player has no incentive to deviate from the Pareto optimal strategy. As a consequence, a Pareto optimum results in the best, collective strategy.

peer 2 can join with an even higher download rate and, therefore, be even more appealing to choose.

#### ***13.5.4 Some BitTorrent details***

We end with additional details concerning BitTorrent. The peer connections are symmetric just as in TCP. Each connection between two peers in a group of peers keeps 2 status bits (a choking-bit and an interested-bit) at each side of the connection. The interested-bit should be kept at all times, also when a downloader needs nothing from the other peer and while being “choked”.

Choking occurs (1) when TCP operates badly, e.g. when the connection suffers from congestion and (2) to enable all peers to execute a “tit-for-tat”-like algorithm to get a fair downloading rate. Further details comprise the standard BitTorrent “choking algorithm” whose specific operation is described in BitTorrent.org (such as optimistic unchoking, the time parameters that indicate how often a peer can choke and unchoke, settings to how many peers uploading can proceed simultaneously, etc.)

### **13.6 Additional aspects**

#### ***13.6.1 Multicast***

All approaches discussed can be extended to multicast<sup>12</sup> (see Section 7.6.4), where the multicast tree is the union of the unicast paths. The performance criteria mentioned above apply as well, together with some multicast-specific criteria (number of links in a tree; degrees of nodes in the tree). The possible freedom of choice for unicast routes can be exploited to optimize the multicast trees.

Two important benchmarks for the performance (e.g. hopcount and delay) of application-layer multicast are native multicast and multiple unicast connections. For simplicity and to a reasonably good approximation in the Internet, we may assume that native unicast and multicast are operating with shortest paths and shortest-path trees in the underlying network. Clearly, native or IP-layer multicast is nearly optimized and, hence, it forms a lower bound for the application layer performance. Multiple unicast as sketched in Fig. 7.17 is a trivial upper bound because no notion of the multicast concept such as the group size and the forwarding along a tree while copying only at the branches, is taken into account. Hence, application-layer multicast is less efficient than native multicast but, especially for large groups, much more efficient than multiple unicast. While the added value of overlay networks is that they facilitate new services and applications with low

---

<sup>12</sup> For example, CAN-MC is based on CAN, Scribe on Pastry and Bayeux on Tapestry.