

Coping with the Parallelism of BitTorrent: Conversion of PEPA to ODEs in Dealing with State Space Explosion

Adam Duguid

Laboratory of Computer Science, The University of Edinburgh
a.j.duguid@sms.ed.ac.uk

Abstract. The Performance Evaluation Process Algebra (PEPA) language is a stochastic process algebra, generating Continuous Time Markov Chains (CTMC) to allow quantitative analysis. Protocols such as BitTorrent are highly parallel in nature, and represent one area where CTMC analysis is limited by the well-known state space problem. The number of unique states each client can exist in, and the number of clients required to accurately model a typical BitTorrent network preclude the use of CTMCs. Recent work has shown that PEPA models also allow the derivation of an activity matrix, from which ODE and stochastic simulation models, as alternative forms of analysis, are possible. Using this technique, a BitTorrent network is created, analysed, and the results compared against previous BitTorrent models.

1 Introduction

The PEPA[1] language originated, in part, from Calculus of Communicating Systems (CCS), allowing the generation of a labelled transition graph with rates based on the exponential distribution. From this graph a CTMC can be obtained and the steady state gained through standard numerical techniques. CTMCs produce exact results, in the sense that every possible state of the system is accounted for and that all probabilities are correct for the given rates. The use of CTMCs assumes that the subsystems behaviour can, to some degree of accuracy, be described with exponential distributions and behaviour is independent of time.

One particular weakness of CTMCs is the size of the model which can be efficiently analysed while still remaining tractable. As the number of components increases, especially components that act independently from one another, the size of the state space can rapidly expand beyond tractable limits. Techniques such as model simplification and state aggregation can allow the analysis of larger models to some extent but the limitations still remain.

Recent work has introduced process algebra (in this instance π calculus) to the area of systems biology[2,3], a field interested in the dynamic pathways of biological systems. However, with the desire to model large numbers of proteins or receptors, the dominant approach within systems biology has been to use

Ordinary Differential Equations (ODEs). ODEs are continuous-time, continuous-state and deterministic in nature. In addition, the area of ODEs is well researched and so supported by a range of solvers. The modelling of biological systems can also be conducted through stochastic simulation such as Gillespie's Stochastic Simulation Algorithm (SSA)[4]. Gillespie's argument for the use of a continuous-time, discrete-state stochastic simulation centres on physical accuracy with the real system, in this case chemical reactions. Both approaches scale differently to CTMCs, where the numbers of each component do not affect the complexity of the model in the same way.

This has led to work mapping PEPA to ODEs[5,6] to model the Extracellular signal Regulated Kinase (ERK) signalling pathway from within a process algebra. Two structures were defined, reagent-centric and pathway-centric, offering different views of the system. Both structures allow the derivation of ODEs from the underlying PEPA model, offering an alternative style of analysis from CTMCs. This same derivation process can be used to allow the use of SSA.

The PEPA language will be briefly covered, followed by a more complete description of the mapping from PEPA to ODEs. The salient parts of the BitTorrent protocol will be detailed, and a PEPA model constructed using the reagent-centric approach. The BitTorrent protocol is used due to its parallel nature of communication between entities, which resists analysis with CTMCs. Attempting to model even 20 peers can easily lead to a state space as large as 100^{20} with CTMCs while remaining tractable when using ODEs or SSAs. Finally, these results will be compared against an existing model of BitTorrent.

2 PEPA

A model defined in the PEPA language consists of a number of components representing different agents or entities in the real system. The components interact with each other through a small set of combinators as shown below.

$$P ::= (\alpha, r).P \mid P + Q \mid P \underset{L}{\bowtie} Q \mid P/L \mid A$$

Prefix $(\alpha, r).P$ represents a component that can perform an activity α at rate r (sampled from the negative exponential distribution) before it transitions to a component of type P .

Choice $P + Q$ represents a component which is either of type P or Q . Which is chosen is based on a race condition on the first activity of each component.

Co-operation $P \underset{L}{\bowtie} Q$ requires that if components P and Q can both perform an activity α (where $\alpha \in L$), then for either component to perform α , they must both perform it together. Where P or Q are capable of an activity not in the set L then these can occur independently.

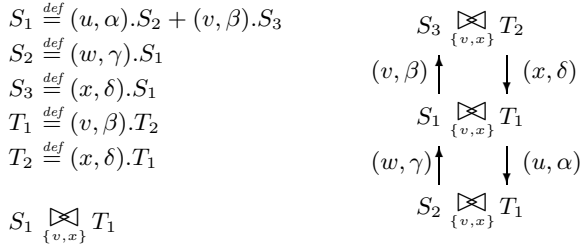


Fig. 1. Toy example highlighting the PEPA language

Hide P/L alters component P such that any activity in the set L is hidden from the rest of the model and cannot be synchronised on.

Constant $A \stackrel{def}{=} P$ assigns names to components.

Figure 1 shows an example in the PEPA language with the choice and co-operation in use. In this example, the component S_1 can freely change into S_2 via the activity u . Activity w can also occur independently. The co-operation over activities v and x mean that both the S_i and T_i components must be in the correct state and both must transition together i.e. $S_1 \bowtie_{\{v,x\}} T_1 \xrightarrow{(v,\beta)} S_3 \bowtie_{\{v,x\}} T_2$ is the only transition possible with activity v . A more complete description of the language can be found in [1], but for the purposes of describing the reagent-centric approach, understanding of the prefix and choice combinators is enough.

2.1 The Reagent-Centric Approach

The reagent-centric approach in its coarsest form defines two states for each component, those being *high* and *low*. Through activities that *consume* the resource or component, the component transitions from a high to low state. Conversely, activities that replenish move a component from a low to high state. This approach is in keeping with the component view of PEPA, where the focus is on the component and the activities possible in that state. The Prefix combinator records the reaction that causes this change, and the rate that the reaction occurs. The Choice combinator allows any one component to be associated with any number of reactions.

Figure 2 shows a small network, and the PEPA reagent-centric model that describes the graphical representation. In this example the PEPA components are A,B and C, and are tagged with H and L to designate the high and low concentrations. By stipulating unique activity names for each reaction, the direction of change (high to low or vice versa) can be used to create a list of reactions with components either being consumed or created through an activity. The PEPA definitions in Fig. 2 give rise to four reactions shown here in the Chemical Model Definition Language (CMDL) $W, X \rightarrow Y, Z$. W is the name for the reaction, $X = \{x_1 + \dots + x_n\}$ lists all the components that are consumed in this reaction. Y is a list in the same format as X representing those

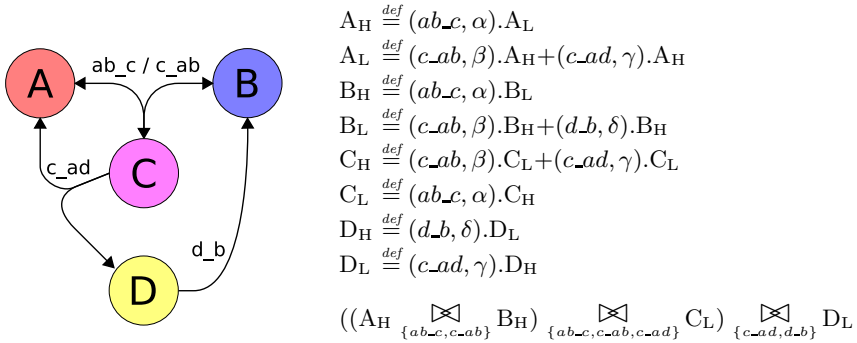
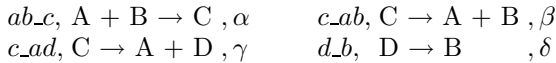


Fig. 2. PEPA reagent-centric example

components increased by this reaction. The last part of the reaction, Z , defines the rate constant from which the reaction rate is derived.



The reaction ab_c consists of two reactants and one product. From the PEPA definition in Fig. 2, it can be seen that both the components A and B can transition from a high to low state via the activity/reaction ab_c , thus placing them as the two reactants of reaction ab_c . Similarly, component C can transition from a low to high state by reaction ab_c placing it as a product of this particular reaction. By iterating through all of the prefixes for each definition, the reactions can be constructed.

The rate at which any one reaction can happen is not simply the defined constant. Where previously the reaction ab_c was defined as $A + B \rightarrow C, \alpha$, we take the constant α and multiply it by the number of molecules in both the A and B components (to allow for permutations of all A molecules interacting with all B molecules) to give a reaction rate of αAB which is known as the mass action rate. The reactions can also take the form of ODEs (as seen below) by a linear transform on the reaction definitions.

$$\begin{array}{ll}
 \frac{dA}{dt} = -\alpha A(t)B(t) + (\beta + \gamma)C(t) & \frac{dC}{dt} = \alpha A(t)B(t) - (\beta + \gamma)C(t) \\
 \frac{dB}{dt} = -\alpha A(t)B(t) + \beta C(t) + \delta D(t) & \frac{dD}{dt} = \gamma C(t) - \delta D(t)
 \end{array}$$

Through these two formats, both stochastic simulation and ODE analysis are available. ODEs derived from PEPA in this manner will always respect the rules of conservation, as PEPA works on a static number of components. The inclusion of stoichiometric information (the quantitative relationship between reactants and products) outside of the PEPA model does however allow for a more powerful representation. Now the numbers of each components required in each reaction are any valid integer i.e. ab_c requires 3 units of component A instead of 1.

3 Modelling a BitTorrent Network

Before a BitTorrent network can be modelled, the salient parts of the protocol must be covered. Prior work by Qui and Srikant shall be examined and contrasted against the intended model before a more in-depth description of the PEPA model.

3.1 The BitTorrent Protocol

Developed in 2001 by Bram Cohen, BitTorrent was designed as a way of distributing the load of hosting a resource, making use of the bandwidth of the users. The BitTorrent protocol encodes information regarding the resource within a torrent file, including SHA-1 hashing of the files and the location of the tracker, where peer discovery occurs. When starting a torrent, the client (at this initial point known as a downloader) must contact the tracker in order to join the ‘swarm’ of active peers. Each contact with the tracker will typically return a randomised peer list of size fifty (where swarm size > 50) and so over time (contact occurring every three to thirty minutes) will represent a well-connected graph.

Once knowledge of other peers is obtained, connections can be made and transfer started. The entire content is split into a number of pieces (one Linux distribution supplies a 2.83GB DVD over 2906 pieces) and the parallelism is in the ability to download these pieces in any order. Using a combination of tit-for-tat, and a set of behaviours for dealing with previously snubbed peers (peers you are currently ignoring), each client attempts to maximise its own downloading speed by uploading to those peers that offer the highest transfer speeds. After each piece is downloaded, it can be offered to other peers instantly.

The splitting of the content into multiple pieces also allows the downloading to happen over greater periods of time. Izal *et al.* had to account for multi-session downloading (where peers disconnect from the swarm and reconnect at a later point, ready to continue where they left off), a feature that is advantageous when dealing with large downloads i.e. 2.83GB operating systems.

The protocol does not enforce a system for peers that are only uploading (known as seeds). The current implementation of BitTorrent by Bram Cohen [7] uses a seeding policy based on uploading to those peers that can download the fastest, the motivation being to create another seed as quickly as possible. As has been noted, this current policy means there is little incentive [8,9] for a client to upload once the entire content has been downloaded.

Lastly, while not part of the protocol, the recommended strategy for piece selection is *rarest-first*. With the exception of the initial piece, rarest-first strategy is used to ensure an even availability of all pieces. Although this only applies within the local group (each peer cannot see availability of pieces beyond those it is connected to) the well-connected property of random graphs will help create an even spread of pieces over time.

3.2 Simple Fluid Model

The fluid model by Qiu and Srikant [10] adopts a high level view of a BitTorrent network, representing peers in one of two states, downloaders or seeders. Consisting of two differential equations (downloaders (1) and seeders (2)), six parameters are used to define the behaviour.

$$\frac{dx}{dt} = \lambda - \theta x(t) - \min\{cx(t), \mu(\eta x(t) + y(t))\} . \quad (1)$$

$$\frac{dy}{dt} = \min\{cx(t), \mu(\eta x(t) + y(t))\} - \gamma y(t) . \quad (2)$$

The main fragment (3), present in both reactions, defines the rate of completion where variable c is the downloading bandwidth, $x(t)$ the number of downloaders at time t , μ the uploading bandwidth, $\eta \in [0, 1]$ the effectiveness of file-sharing and $y(t)$ the number of seeds at time t . This fragment simply states that the rate peers can complete the download is defined by either the speed the peers can download at or the speed at which all peers are uploading, whichever is smaller. The other variables are γ for seed disconnect rate, λ for peer arrival rate and θ for termination of downloading.

$$\min\{cx(t), \mu(\eta x(t) + y(t))\} . \quad (3)$$

In the paper, η is defined as $\eta \approx 1 - \left(\frac{\log N}{N}\right)^k$ where N is the number of pieces and k is the number of other downloaders currently connected to. If we let $N = 2906$, as seen for the DVD example, even where $k = 1$ $\eta \approx 0.9988$. Additionally, the upload capacity of the typical consumer is smaller than the download capacity available, thus $\frac{c}{\mu} \geq 1$ and with typical asynchronous connections (2Mbps download, 512Kbps upload) $\frac{c}{\mu} \geq 4$. This can be used to approximate (3) to $\mu(x(t) + y(t))$ when $x(t)\left(\frac{c}{\mu} - 1\right) > y(t)$, a condition satisfied for the majority of the time in Qiu and Srikant's own results. Unsurprisingly it can be seen from this approximation that the rate of change from downloader to seeder within the model will follow the exponential distribution with rate μ but more importantly, with $\sigma^2 = \frac{1}{\mu^2}$.

Instead, the PEPA model will compartmentalise the downloading action over one hundred steps, acting as a percentage complete indicator. To accurately model a BitTorrent network all permutations of the pieces should be recorded. As this would require 2^N states (and so intractable for any but the most trivial values of N) the approximation to one hundred states for percentage complete is deemed adequate. This action can be represented by the Erlang distribution (Erlang CDF F_e shown in (4)) and the effect of splitting the download into k compartments seen in Fig. 3.

$$F_e = 1 - e^{-\lambda x} \sum_{n=0}^{k-1} \frac{(\lambda x)^n}{n!} \quad \text{where } k = \text{number of compartments,} \quad (4)$$

and $\lambda = \text{rate for each compartment} .$

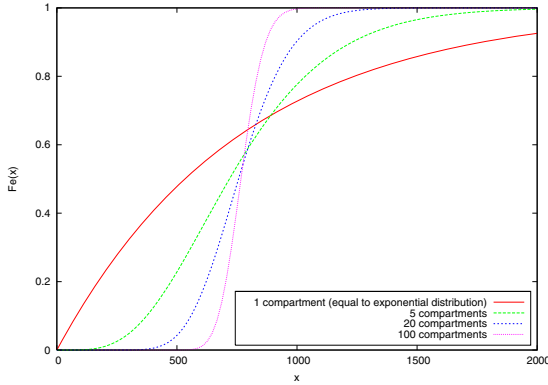


Fig. 3. CDF for the Erlang distribution. The effect of splitting a single state change ($\lambda = 0.0013$) to k compartments.

With the Erlang distribution, $\bar{x} = \frac{k}{\lambda}$ and $\sigma^2 = \frac{k}{\lambda^2}$, so if we define $\lambda' = k\lambda$ the means of the original distribution and the Erlang distribution are identical ($\frac{1}{\lambda} = \frac{k}{\lambda'}$) but the variance is reduced by a factor of k ($\frac{1}{\lambda^2}$ compared to $\frac{k}{(\lambda')^2} = \frac{1}{k\lambda^2}$), an important consideration when dealing with events with such large delays.

While the fluid model defined by Qui and Srikant could undoubtedly be modified to follow the Erlang distribution, the advantage of explicitly defining the one hundred steps is to accommodate other observable peer behaviour. Besides allowing a peer to disconnect, the PEPA model will incorporate the concept of multi-session downloads [8], where a peer can temporarily disconnect from the network, before rejoining and continuing from where it left off.

3.3 The PEPA Model

For the PEPA reagent-centric model, the following assumptions are made:

1. The network shall consist of only homogeneous, asynchronous connections.
2. The source for the content (original seeder) will have a persistent online presence. The torrent in this instance cannot die (less than 100% of the content being available).
3. Through the use of randomised peer lists and tit-for-tat the upload of the peer can be fully utilised if there is demand.
4. Through the use of randomised peer lists, rarest first piece selection and assumption 2, all pieces are available to all peers.
5. Peer behaviour is independent of time.

Assumptions 1 and 5 are gross simplifications of the real system but allow for a tractable model. Modelling different types of connections can have a multiplicative effect on the number of states, as every action may be influenced by different types of peer. Peer behaviour is unlikely to be independent of time or even consistent across peers. Pouwelse *et al.* [9] found that new torrents were

often accompanied by a huge surge of downloaders very early on, labelling this the *flashcrowd effect*.

Assumption 2 is justified by the expected use of the BitTorrent protocol. Where content providers previously relied on HTTP or FTP, the content was required to be permanently accessible. This would not change with the use of the BitTorrent protocol. The content owners may optimise such that with enough seeds the original is not active, but to maintain availability it would be required to seed again if the number of seeds dropped below some threshold.

Assumptions 3 and 4 are outwith the scope of this paper. They rely on the properties of random graphs, the small world assumption and epidemiology. Without these assumptions, modelling the BitTorrent protocol becomes increasingly more complex. Of these assumptions, only 2 is not also made by Qui and Srikant.

Using these assumptions, the model will incorporate certain behaviours, these being:

1. Each peer within the swarm shall be either a downloader or a seed.
2. A downloader will be split into one hundred states, to record the level of completion.
3. Both downloaders and seeds can quit or go offline at any stage, returning online will return them to their previous state.

These properties can be translated into four groups of definitions within the PEPA reagent-centric approach, with one group for each of the peer states (online, offline, downloading and seeding). Two additional components, `upload_pool` and `peer_deac`, are used to control behaviour of the model. The existence of the downloading state is to enforce correct behaviour regarding use of the available upload bandwidth.

$$\text{Online_Peer-0}_H \stackrel{\text{def}}{=} (\text{connect}_0, \text{connect_rate}).\text{Online_Peer-0}_L + \\ (\text{offline}_0, \text{offline_rate}).\text{Online_Peer-0}_L + \\ (\text{quit}_0, \text{quit_rate}).\text{Online_Peer-0}_L$$

$$\text{Online_Peer-0}_L \stackrel{\text{def}}{=} (\text{torrent}, \text{torrent_rate}).\text{Online_Peer-0}_H + \\ (\text{online}_0, \text{online_rate}).\text{Online_Peer-0}_H$$

$$\text{Online_Peer-}n_H \stackrel{\text{def}}{=} (\text{connect}_n, \text{connect_rate}).\text{Online_Peer-}n_L + \\ (\text{offline}_n, \text{offline_rate}).\text{Online_Peer-}n_L + \\ (\text{quit}_n, \text{quit_rate}).\text{Online_Peer-}n_L$$

$$\text{Online_Peer-}n_L \stackrel{\text{def}}{=} (\text{downloaded}_n, \text{d_rate}).\text{Online_Peer-}n_H + \\ (\text{online}_n, \text{online_rate}).\text{Online_Peer-}n_H \\ n \in \{1 \dots 99\}$$

Here we have the online peer PEPA definitions. The activities quit_n and offline_n cause the levels of an online peer to decrease, seen as activities that change a resource from high to low. Conversely the activity online_n increases the numbers of that particular peer. New peers enter the system at `Online_Peer-0`, through the *torrent* activity and enter the ready state for downloading via the connect_n activity. As entering the downloading state is a change of state, the connect_n

activity reduces the number of online peers for that given state. Lastly, the act of completing another part of the download via the $downloaded_n$ activity increases the number of online peers for that percentage.

$$\begin{aligned}
 \text{Downloading_Peer-}n_H &\stackrel{\text{def}}{=} (\text{downloaded}_{n+1}, d_rate).\text{Downloading_Peer-}n_L + \\
 &\quad (\text{downloading_offline}_n, \text{offline_rate}).\text{Downloading_Peer-}n_L + \\
 &\quad (\text{downloading_quit}_n, \text{quit_rate}).\text{Downloading_Peer-}n_L \\
 \text{Downloading_Peer-}n_L &\stackrel{\text{def}}{=} (\text{connect}_n, \text{connect_rate}).\text{Downloading_Peer-}n_H \\
 \text{Offline_Peer-}n_H &\stackrel{\text{def}}{=} (\text{online}_n, \text{online_rate}).\text{Offline_Peer-}n_L \\
 \text{Offline_Peer-}n_L &\stackrel{\text{def}}{=} (\text{offline}_n, \text{offline_rate}).\text{Offline_Peer-}n_H + \\
 &\quad (\text{downloading_offline}_n, \text{offline_rate}).\text{Offline_Peer-}n_H \\
 n &\in \{0 \dots 99\}
 \end{aligned}$$

Next are the PEPA definitions for the downloading and offline states. The first defined activity for a $\text{Downloading_Peer}_n$ is $downloaded_{n+1}$. Here we can see how progression through the states happens. The number of downloading peers with $x\%$ complete decreases when they manage to complete another percent, taking them to $x+1\%$. As already seen in the online peer definition, the number of peers with $x\%$ complete increases through the $downloaded_x$ activity, and so already some of the behaviour of the model can be seen.

The downloading peer state like the online peer can quit or go offline. The d_rate represents the length of time to download 1% of the torrent and can easily be several minutes in duration, and so this is the state where peers will spend the most time. Without the ability to terminate in this state the rates for quitting and going offline are skewed as $x(t) = \sum_i \text{Online_Peer}_i + \text{Downloading_Peer}_i$.

$$\begin{aligned}
 \text{Seed}_H &\stackrel{\text{def}}{=} (\text{seed_quit}, \text{seed_quit_rate}).\text{Seed}_L + (\text{seed_offline}, \text{seed_offline_rate}).\text{Seed}_L \\
 \text{Seed}_L &\stackrel{\text{def}}{=} (\text{downloaded}_{100}, \text{downloaded_rate}).\text{Seed}_H + \\
 &\quad (\text{seed_online}, \text{seed_online_rate}).\text{Seed}_H \\
 \text{Offline_Seed}_H &\stackrel{\text{def}}{=} (\text{seed_online}, \text{seed_online_rate}).\text{Offline_Seed}_L \\
 \text{Offline_Seed}_L &\stackrel{\text{def}}{=} (\text{seed_offline}, \text{seed_offline_rate}).\text{Offline_Seed}_H
 \end{aligned}$$

As can be seen, the seed plays a passive role, where simply their presence (or lack of) is the only information required.

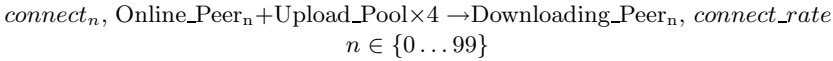
The control states Upload_Pool and Peer_deac are designed to maintain the upload capacity within the network. Upload_Pool acts as a counter for unused upload bandwidth while Peer_deac was created to prevent skew to the rates at which seeds and peers would quit or go offline. For each seed or online peer that disconnects from the swarm, the upload pool will be decremented. The rate at which seeds and peers disconnect though should not depend on the current level of the upload pool while conservation tells us we can not have populations of negative values. Instead the Peer_deac acts as a reservoir, holding all the disconnect requests and reducing the upload pool as and when possible. This will become clearer when the PEPA is converted.

$$\begin{aligned}
\text{Upload_Pool}_H &\stackrel{\text{def}}{=} (\text{connect}_n, \text{connect_rate}).\text{Upload_Pool}_L + \\
&\quad (\text{deallocation}, \text{deallocation_rate}).\text{Upload_Pool}_L \\
\text{Upload_Pool}_L &\stackrel{\text{def}}{=} (\text{downloaded}_p, \text{d_rate}).\text{Uploaded_Pool}_H + \\
&\quad (\text{online}_n, \text{online_rate}).\text{Upload_Pool}_H + \\
&\quad (\text{downloading_offline}_n, \text{offline_rate}).\text{Uploaded_Pool}_H \\
&\quad (\text{downloading_quit}_n, \text{quit_rate}).\text{Uploaded_Pool}_H \\
&\quad (\text{seed_online}, \text{seed_online_rate}).\text{Upload_Pool}_H \\
&\quad n \in \{0 \dots 99\}, p \in \{1 \dots 100\} \\
\text{Peer_deac}_H &\stackrel{\text{def}}{=} (\text{deallocation}, \text{deallocation_rate}).\text{Peer_deac}_L \\
\text{Peer_deac}_L &\stackrel{\text{def}}{=} (\text{offline}_n, \text{offline_rate}).\text{Peer_deac}_H + \\
&\quad (\text{quit}_n, \text{quit_rate}).\text{Peer_deac}_H + \\
&\quad (\text{seed_offline}, \text{seed_offline_rate}).\text{Peer_deac}_H + \\
&\quad (\text{seed_quit}, \text{seed_quit_rate}).\text{Peer_deac}_H \\
&\quad n \in \{0 \dots 99\}
\end{aligned}$$

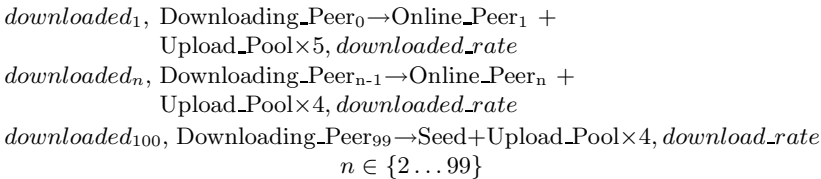
The PEPA model defined allows the following behaviour. Peers join the swarm with 0% of the content complete and without the ability to upload. They can enter a state ready to download (resources committed) and increase the percentage complete at which point the allocated resources are released. Peers with 1 to 100% complete contribute to the upload capacity. All peers (online, downloading and seeds) can enter an offline state or quit where those that had contributed to the upload (1 to 100%) reduce the upload capacity.

3.4 Reaction Based System

With BitTorrent modelled in the reagent-centric style, it can be translated into a set of reactions suitable for either stochastic simulation or analysis through ODEs. Based on the PEPA definitions, the connect_n activity allocates the required upload bandwidth and changes an online peer to a downloading peer. As previously stated, the use of stoichiometric information is external to the PEPA model, but can be clearly seen here.



Here the use of the Upload_Pool can be clearly seen. Once a peer has part of the content, they return the previously allocated resources, plus an additional unit that they now contribute to the swarm. Once the last part of the content is downloaded a downloading peer can be seen to transition to a seed



Within the *offline* activity, the influence of the Peer_deac is evident. Here, and in the definitions for seeds going offline, the delay in updating the upload pool

allows the activity to happen at the correct rate. At a later point the *deallocation* activity will decrement the upload pool as required.

$$\begin{aligned} \text{offline}_0, & \quad \text{Online_Peer}_0 \rightarrow \text{Offline_Peer}_0 + \text{Peer_deac}, \text{offline_rate} \\ \text{offline}_n, & \quad \text{Online_Peer}_n \rightarrow \text{Offline_Peer}_n + \text{Peer_deac}, \text{offline_rate} \\ & \quad n \in \{1 \dots 99\} \end{aligned}$$

$$\text{deallocation}, \text{Upload_Pool} + \text{Peer_deac} \rightarrow \text{, deallocation_rate}$$

The behaviour of a downloading peer that goes offline or quits is different to that of an online peer. The reason is the use of `Peer_deac` is not required. The previous securing of resources accounts for more than any one peer contributes to the system. Thus a downloading peer only has to relinquish one unit less than it reserved. This allows the model to behave correctly without additional use (and minor delays) of the deallocation counter.

$$\begin{aligned} \text{downloading_offline}_0, & \quad \text{Downloading_Peer}_0 \rightarrow \text{Offline_Peer}_0 + \text{Upload_Pool} \times 4, \text{offline_rate} \\ \text{downloading_offline}_n, & \quad \text{Downloading_Peer}_n \rightarrow \text{Offline_Peer}_n + \text{Upload_Pool} \times 3, \text{offline_rate} \\ & \quad n \in \{1 \dots 99\} \end{aligned}$$

The online_n definitions allow peers that went offline to return to the swarm. As can be seen, if returning from a partial download state their return increments the upload capacity within the network.

$$\begin{aligned} \text{online}_0, & \quad \text{Offline_Peer}_0 \rightarrow \text{Online_Peer}_0, \text{online_rate} \\ \text{online}_n, & \quad \text{Offline_Peer}_n \rightarrow \text{Online_Peer}_n + \text{Upload_Pool}, \text{online_rate} \\ & \quad n \in \{1 \dots 99\} \end{aligned}$$

Similar rules exist for the seeds. The only difference lies within the rate at which seeds can leave the system. To enforce the second assumption of the BitTorrent network, the rate is changed from $\text{seed_online_rate} \times \text{Seed}$ to $\text{seed_online_rate} \times (\text{Seed} - 1)$ and so enforces the continued existence of one seed at all times.

4 Analysis

Of the three experiments performed by Qui and Srikant, the third compared the fluid model against log files from a real torrent file (the file in question was 530MB in size). Using these logs, they derived values for the six parameters with λ and γ found to vary with time. Simplified slightly, the parameters used in the comparison were $\eta = 1$, $\theta = 0.001$, $\mu = 0.0013$ and $c = 1$. As already stated λ and γ varied with time and so were set at $\lambda = 0.06$ and $\gamma = 0.001$ for $t \leq 800$, and $\lambda = 0.03$ and $\gamma = 0.0044$ for $t > 800$.

For the PEPA model to mirror the fluid model, the ability of the downloaders and seeds to switch between online and offline was turned off. To approximate the time dependent arrival rate, a linear function was fitted to the data and calculated with $\text{torrent_rate} = 0.0368577$. As the dynamic seed quit rate (γ) is connected to seed population it can not be fitted in the same way and hence

fixed at 0.0044. Lastly, the download rate was normalised to account for the content being downloaded in 1% chunks and that enough upload capacity was allocated to maximise each peer, leaving the $downloaded_rate = 0.52$.

The results for these two models can be seen in Fig. 4. The peaks connected to the fluid model are the effect of the dynamic parameters. The effect of cascade of exponential distributions can be easily seen in the delay before a new seed can appear within the PEPA model. Whilst the steady state value for the seeds is similar between the two models, the value for the downloaders is noticeably different, a potential cause for concern given the low values encountered. Figure 5 compares the same PEPA model against a fluid model with non dynamic parameters, λ and γ being fixed to those values used within the PEPA model. From Fig. 4 a difference in the number of downloaders within the model still exists but is now reduced, which can be explained by fixing the rate at which new peers enter the system. When using fixed parameters, the steady state values for downloaders and seeds are reasonably close between the two models.

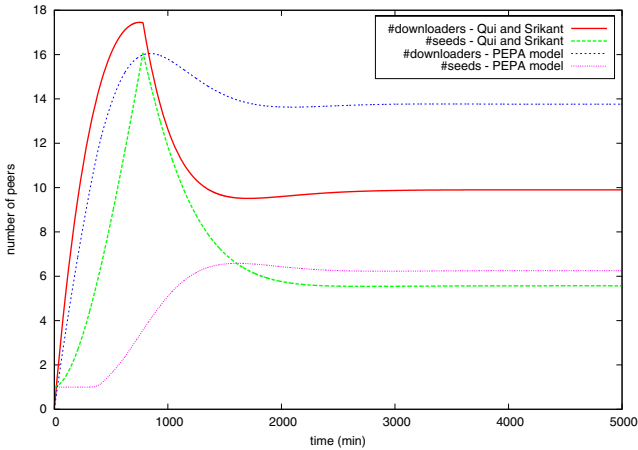


Fig. 4. Comparison of fluid and PEPA model (fluid model using time varying parameters)

To further contrast the two models, a second experiment was run, where the size of the file was increased. The size of the file was set to 2.83GB (the DVD previously mentioned) and the peers assumed to have a 2Mbps download capacity and 256Kbps upload. This capacity equates to $\mu = \frac{1}{1648}$ and $downloaded_rate = \frac{1}{4}$, while leaving c alone does not affect the model. All other parameters were left at their original (fixed) values. Figure 6 shows the results for this. This experiment accentuates the differences seen within the first experiment. While the number of seeds in both models is approximately the same, the difference in the number of downloaders has increased.

The final graph, Fig. 7 highlights the flexibility built into both the use of PEPA as the modelling language and of this particular model. Gillespie's Stochastic

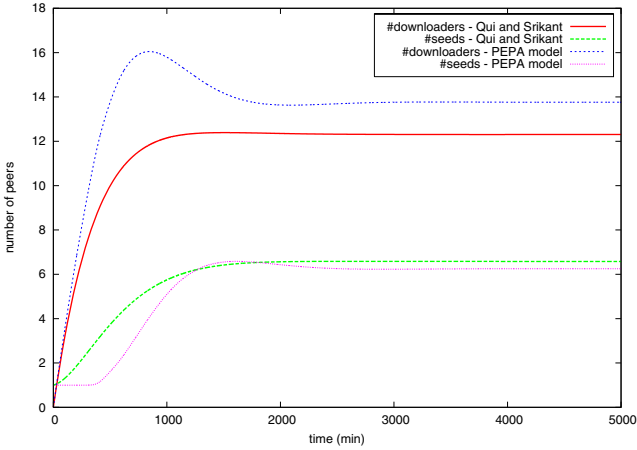


Fig. 5. Comparison of fluid and PEPA model (fluid model using fixed parameters)

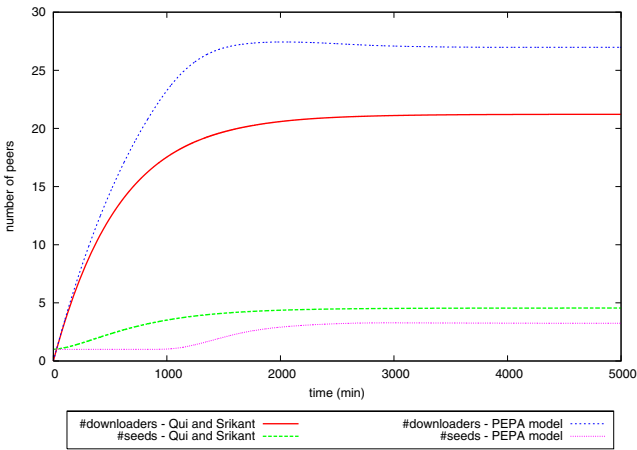


Fig. 6. PEPA and fluid model with larger files

Simulation Algorithm (SSA) was run over five independent replications as an alternative to use of ODEs. Where stochastic noise may be of interest, or the population sizes of the components are low, the use of stochastic simulations may prove advantageous. For the comparisons against the fluid model, the ability to change from online to offline was inhibited. Here, with exaggerated parameters, the effects of the offline state can be seen (more so with the downloaders). By shifting a certain percentage of the downloaders from the active state, the available bandwidth is reduced causing a more shallow gradient towards the steady state level.

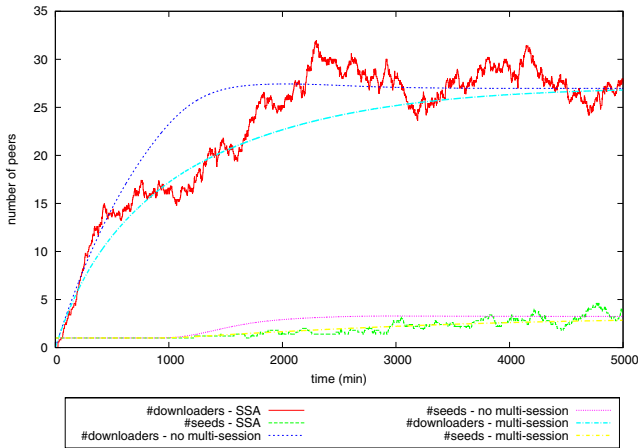


Fig. 7. PEPA model with and without multi-session ability and stochastic simulation

5 Conclusions

The ability to convert PEPA models to a reaction based system, allowing the use of ODE analysis or stochastic simulation now allows models previously too large for more traditional forms of analysis. Using the BitTorrent protocol to highlight this fact, a model where a peer can exist in any one of three hundred states was created and compared against a simple two state fluid model. Under CTMCs this would have created a potential state space of 300^x where x is the number of peers. The PEPA reagent-centric approach also affords a cleaner representation whilst providing access to both deterministic and stochastic solutions. As the size of the model increases, the complexity of the ODEs can hinder understanding of the system or alteration if required.

Whilst the translation process is now well defined, the mapping of CTMC analysis to time-series analysis is still work in progress. Regardless, the ability to convert to a reaction based system is a useful tool.

The BitTorrent model itself has shown that not only can a large system be defined within the PEPA language, but also offered a more detailed view of a BitTorrent network. It expanded on the simple fluid model, detailing many of the extra states. Many of the parameters require assigning based on real world data, the next step being to obtain records from a tracker. It should also be noted that the full flexibility of the model has not been covered. While one of the assumptions was that peer behaviour was independent of time, the alteration of rates for different levels of completion would allow a certain level of controllable behaviour, i.e. a peer is more likely to disconnect in the first five percent and extremely unlikely in the last ten percent. Again, this data can be gathered by obtaining records from a tracker.

References

1. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press (1996)
2. Regev, A., Silverman, W., Shapiro, E.: Representation and simulation of biochemical processes using the pi-calculus process algebra. In: Proceedings of the Pacific Symposium of Biocomputing (PSB2001). (2001) 459–470
3. Priami, C., Regev, A., Silverman, W., Shapiro, E.: Application of a stochastic name passing calculus to representation and simulation of molecular processes. *Information Processing Letters* **80** (2001) 25–31
4. Gillespie, D.: Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry* **81**(25) (1977) 2340–2361
5. Calder, M., Gilmore, S., Hillston, J.: Modelling the influence of RKIP on the ERK signaling pathway using the stochastic process algebra PEPA. In: Proceedings of BioConcur'04, London, England (2004) Extended version to appear in *Transactions on Computational Systems Biology*.
6. Calder, M., Gilmore, S., Hillston, J.: Automatically deriving ODEs from process algebra models of signalling pathways. In: Proceedings of Computational Methods in Systems Biology '05, Edinburgh, Scotland (2005)
7. Cohen, B.: Incentives Build Robustness in BitTorrent. <http://www.bittorrent.com/bittorrentecon.pdf> (2003)
8. Izal, M., Urvoy-Keller, G., Biersack, E.W., Felber, P., Hamra, A.A., Garcés-Erice, L.: Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In Barakat, C., Pratt, I., eds.: PAM. Volume 3015 of Lecture Notes in Computer Science., Springer (2004) 1–11 <http://www.pam2004.org/papers/148.pdf>.
9. Pouwelse, J., Garbacki, P., Epema, D., Sips, H.: The BitTorrent P2P File-sharing System: Measurements and Analysis. In Castro, M., van Renesse, R., eds.: IPTPS. Volume 3640 of Lecture Notes in Computer Science., Springer (2005) http://www.st.ewi.tudelft.nl/~pouwelse/Bittorrent_Measurements_6pages.pdf.
10. Qiu, D., Srikant, R.: Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In Yavatkar, R., Zegura, E.W., Rexford, J., eds.: SIGCOMM, ACM (2004) 367–378 http://comm.csl.uiuc.edu/srikant/Papers/sigcomm04_revised.pdf.